Techniques for Developing an Acquisition Strategy by Profiling Software Risks

Mary Catherine Ward
Joseph P. Elm
Susan Kushner

August 2006

TECHNICAL REPORT
CMU/SEI-2006-TR-002
ESC-TR-2006-002

Carnegie Mellon
Software Engineering Institute
Pittsburgh, PA 15213-3890

Techniques for Developing an Acquisition Strategy by Profiling Software Risks

CMU/SEI-2006-TR-002
ESC-TR-2006-002

Mary Catherine Ward
Joseph P. Elm
Susan Kushner

August 2006

Acquisition Support Program

This report was prepared for the

SEI Administrative Agent
ESC/XPK
5 Eglin Street
Hanscom AFB, MA 01731-2100

The ideas and findings in this report should not be construed as an official DoD position. It is published in the interest of scientific and technical information exchange.

This work is sponsored by the U.S. Department of Defense. The Software Engineering Institute is a federally funded research and development center sponsored by the U.S. Department of Defense.

For information about purchasing paper copies of SEI reports, please visit the publications portion of our Web site (http://www.sei.cmu.edu/publications/pubweb.html).

Table of Contents

List of Figures

List of Tables

Acknowledgements

In developing this report, several individuals made invaluable contributions regarding its content and clarity. Specifically, we want to thank our Carnegie Mellon® Software Engineering Institute (SEI) colleagues for their foundational research: Cecilia Albert, John Bergey, Wolfhart Goethert, and Ed Morris. Julie Cohen, Stephen Blanchette, Jr., Brian Gallagher, Mary Popeck, and Eileen Wrubel provided us with insightful feedback. Special thanks go to Thomas Miller for relentlessly testing the functionality and checking the consistency of the Acquisition Strategy Development Tool. Rachel Callison of the SEI Library was instrumental in tracking down elusive reference information. Also, Barbara White of the Communications department demonstrated her Web-savvy by creating the download form for the Acquisition Strategy Development Tool.

Finally, we would like to thank Robert Carnevale of U.S. Army Ground Combat Command and Control for piloting aspects of this research. The support of the U.S. Army Strategic Software Improvement Program (ASSIP) and Dr. James Linnehan of the Office of the Assistant Secretary of the Army for Acquisition, Logistics, and Technology made this work possible.

® Carnegie Mellon is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

Executive Summary

As shown in Figure 1, the act of acquisition planning is one of defining and maintaining an overall approach for a program. The acquisition planning process guides all elements of program execution to transform the mission need into a fielded system that is fully supported and delivers the desired capability. The goal of acquisition planning is to provide a roadmap that will be followed to maximize the chances of successfully fielding a system that meets users' needs within cost and on schedule. Acquisition planning is an iterative process; feedback loops impact future acquisition planning activities.

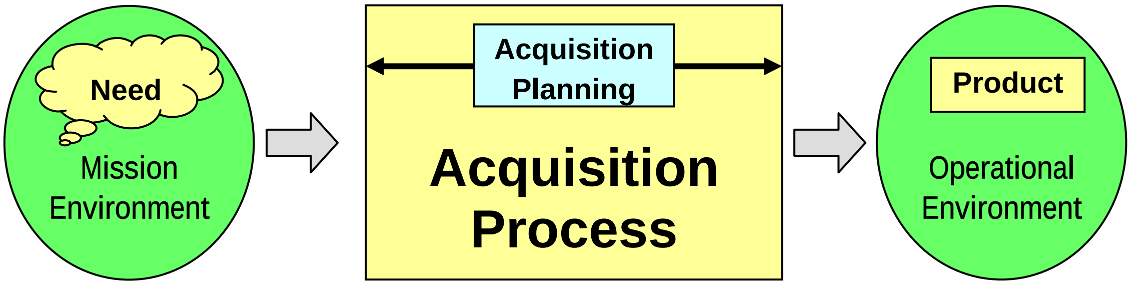


Figure 1: The Context of Acquisition Planning

Many programs struggle during acquisition planning because even though a wealth of information exists, the process itself is not clearly defined or understood. Acquisition planning is a powerful process that can help define the most promising acquisition options that best mitigate risks. In turn, developing an acquisition strategy is a key component of acquisition planning that provides a means of addressing program risks through the program structure. In most cases, however, acquisition planning and acquisition strategy development are done in an ad hoc manner, without consideration of the internal and external factors driving the project. When developing an acquisition strategy, it is important to understand these factors because they often correlate to what drives the program's risk.

For software-intensive systems, the acquisition strategy must acknowledge the type and extent of software risk to the program and include plans to actively mitigate those risks. Programs need structured ways to reason about software risks, formulate acquisition strategies to mitigate software risk, and evaluate their current acquisition strategy in an ongoing, systematic manner.

The acquisition strategy guides decisions made throughout the life cycle of the program. It should be based on reasoning about both internal and external complex factors with regard to the system. When developing an acquisition strategy, it is important to understand the program's driving factors. These *drivers* are conditions that impact program risk. For example, the skills of the personnel working in the program office, the number of stakeholders, and the number of system configurations that need to be maintained are all drivers. Drivers can influence the acquisition strategy independently or there can be interactions between drivers that affect program risk. For example, unstable requirements usually influence overall costs.

Figure 2 shows the relationships among the program's drivers, its risks, and its acquisition strategy.

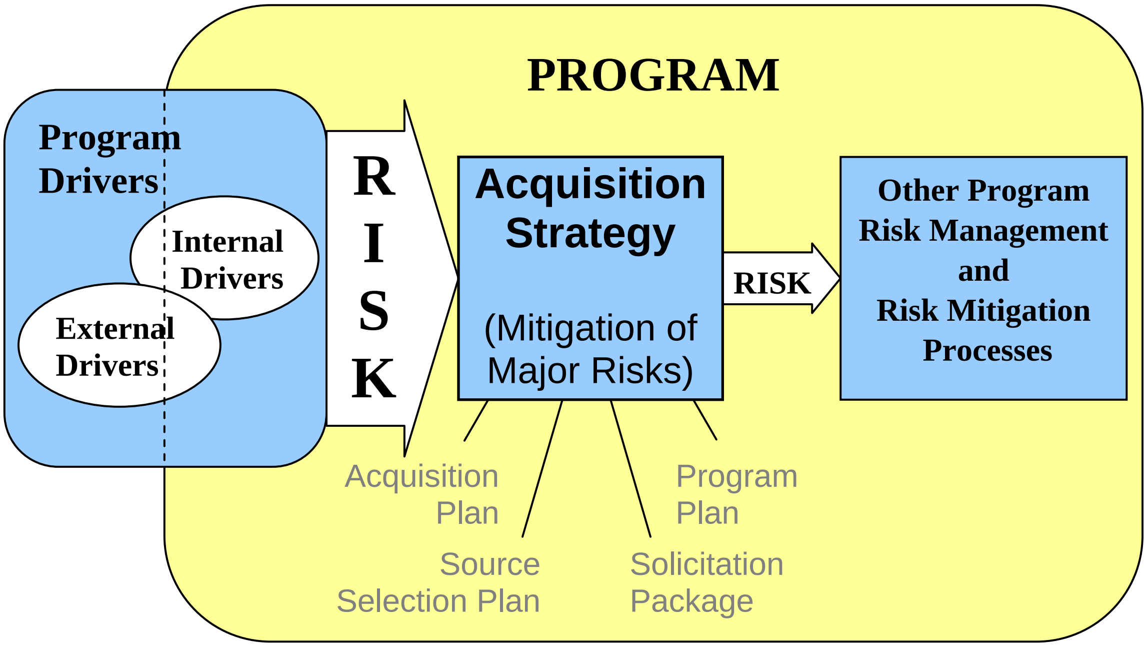


Figure 2: Relationships Among Drivers, Risks, and Acquisition Strategy

The research results presented in this report support a more systematic approach to reasoning about software risk on a program. The methods and techniques presented contribute to the work that focuses on developing an acquisition strategy from a sound, systems engineering approach.

This report outlines research performed to help acquisition planners more systematically

- profile conditions that impact program risk (drivers)
- identify sources of potential software risk early in the program and throughout the program life cycle
- develop an acquisition strategy by decomposing it into the strategy elements that compose it, addressing them individually and then collectively
- reason about their acquisition strategy's ability to mitigate software risk in an ongoing manner

The report introduces a taxonomy of strategy drivers and strategy elements. It also provides a method for performing a comparative analysis of the strategy drivers and the resulting strategic choices for the elements. For a program with an acquisition strategy already in execution, the method can be used to perform a strategy analysis. The method uses a technique, slider bars, to support a more systematic approach to reasoning about software risk on a program.

For example, to use this method and the slider bar technique to develop a strategy, acquisition planners would perform the following steps:

1. Define the objectives of the acquisition.
2. Identify and evaluate the factors that drive the program.
3. Decompose the strategy into individual strategy elements.
4. For the selected strategy elements, identify the potential strategic choices and rank them in order of their risk mitigation capabilities for your program.
5. Evaluate the strategy drivers for the program to identify those that influence the strategy element through the introduction of risk that may be mitigated by the strategy element.
6. Define the relationship between the risk generated by the strategy driver and the risk mitigation capabilities of the strategy element.
7. Map the driver evaluations from Step 2 to the strategy element using the slider bars.
8. Choose the strategy that best mitigates the risk elements.
9. Identify residual risks.

The Acquisition Strategy Development Tool (ASDT) is a customized Excel workbook we created to help acquisition planners work through the method and techniques. The ASDT is provided so that acquisition organizations do not have to develop slider bar templates from scratch. The ASDT is available for download on the SEI Publications Web site at http://www.sei.cmu.edu/publications/publications.html. Note that acquisition planners can also apply the method and techniques discussed in this report without using the ASDT.

Our research focused on identifying and mitigating software risk in a program during the acquisition planning process. However, the methods and techniques we present can be applied to acquisition planning areas other than software risk.

Abstract

The goal of acquisition planning is to create a roadmap that a program can follow to maximize its chances of successfully fielding a system that meets users' needs within cost and on schedule. Developing an acquisition strategy is a key component of acquisition planning that provides a means of addressing risks through the program structure. Programs need structured ways to reason about software risks, formulate acquisition strategies to mitigate software risk, and evaluate their current acquisition strategy in an ongoing, systematic manner.

This report introduces a taxonomy of strategy drivers and strategy elements and provides a method for performing a comparative analysis of the strategy drivers and the resulting strategic choices for the elements. The primary audience for this technical report and the accompanying Excel-based tool is program managers of government acquisition programs. The main prerequisite for successfully using this information is a working knowledge of government acquisition practices.

1 Introduction

1.1 Purpose

The goal of acquisition planning is to provide a roadmap that will be followed to maximize the chances of successfully fielding a system that meets users' needs within cost and schedule. An acquisition plan guides all elements of program execution to transform the mission need into a fully supported, fielded system that delivers the desired capability.

Many programs struggle during acquisition planning because even though a wealth of information exists, the process itself is not clearly defined or understood. Software acquisition planning is a powerful process that can help define the most promising acquisition options that best mitigate risks in the development of software. Developing an acquisition strategy is a key component of acquisition planning that provides a means of addressing program risks through the program structure. In most cases, however, acquisition planning and acquisition strategy development is done in an ad hoc manner, without consideration of the internal and external factors driving the project. When developing an acquisition strategy, it is important to understand these factors because they often correlate to what drives the program's risk.

For software-intensive systems, the acquisition strategy must acknowledge the type and extent of software risk to the program and include plans to actively mitigate those risks. Programs need structured ways to reason about software risks, formulate acquisition strategies to mitigate software risk, and evaluate their current acquisition strategy in an ongoing, systematic manner.

During the past year, the Acquisition Support Program at the Carnegie Mellon® Software Engineering Institute researched more systematic approaches to reason about software risk on a program. Specifically, we explored answers to the questions "Are the methods and techniques arising from a sound, systems engineering approach ones that can be applied to acquisition planning and acquisition strategy development? Will they enable programs to better reason about software risks, formulate acquisition strategies to mitigate those risks, and evaluate their current strategy in an ongoing, systematic manner?"

This report outlines the research that was performed and is designed to help acquisition planners more systematically

® Carnegie Mellon is registered in the U. S. Patent and Trademark Office by Carnegie Mellon University.

- profile conditions that impact program risk (drivers)
- identify sources of potential software risk early in the program and throughout the program life cycle
- develop an acquisition strategy by decomposing it into the strategy elements that compose it, addressing them individually and then collectively
- reason about their acquisition strategy's ability to mitigate software risk in an ongoing manner

The report introduces a taxonomy of strategy drivers and strategy elements. It also provides a method for performing a comparative analysis of the strategy drivers and the resulting strategic choices for the elements. The method proposed is a bi-directional, graphical method of examining and analyzing the relationships between the drivers and the strategic choices. The method is bi-directional in that it enables a program to

- analyze strategy drivers and choose strategies for each strategy element to minimize the risks induced by those drivers, or
- choose a strategy for each strategy element and analyze the risks induced by the program's strategy drivers

The method uses a technique, slider bars, to support a more systematic approach to reasoning about software risk on a program. This work extends a concept introduced by the Department of the Navy representing strategy elements as a continuum or sliding scale, much in the way the volume control on a stereo can range from soft to loud [Department of the Navy 01]. Slider bars can be used to graphically visualize a program's strategy drivers, acquisition strategy element strategic choices, and the relationships among the drivers and choices. The Acquisition Strategy Development Tool (ASDT) explained in Section 1.3 is provided to assist program offices work through the methods and techniques.

Our research focused on identifying and mitigating software risk in a program during the acquisition planning process. However, the methods and techniques we present can be applied to acquisition planning areas other than software risk.

1.2 Document Structure

Section 2 discusses the need for acquisition planning and the roles of the acquisition strategy and acquisition plan in the acquisition planning process. It defines key terms and highlights the relationship between acquisition planning and software acquisition planning including the challenges posed by software.

Section 3 introduces a taxonomy of strategy drivers to help programs identify, evaluate, and understand acquisition strategy drivers. It discusses each strategy driver to assist program personnel to assess their program's software acquisition risk.

Section 4 discusses acquisition strategies in more detail and sample acquisition strategy elements including Acquisition Approach, Competition, Solicitation, Contract Approach, Training, and Source of Support. This is not a comprehensive set of strategy elements; it is only a subset used to illustrate the acquisition strategy development process proposed in this report.

Section 5 provides a method for performing a comparative analysis of the strategy drivers and the strategic choices for each strategy element. The method uses a technique, slider bars to support a more systematic approach to reasoning about software risk on a program. This section describes how to profile a program's strategy drivers, its acquisition strategy elements and corresponding strategic choices, and the relationship between the drivers and strategic choices for each element.

Appendix A provides a hard copy of the template used in the ASDT. This tool is discussed further in Section 1.3. The template can be used to profile the program's key strategy drivers. It can also be used to identify specific strategy element choices and evaluate how those choices mitigate the program's software risks.

1.3 Acquisition Strategy Development Tool

The purpose of the ASDT is to help government program offices formulate and evaluate how their acquisition strategies address the major software risks faced by the program in a more systematic way. It can be used to profile the program's software acquisition characteristics and identify key strategy drivers. It can also be used to identify specific strategic choices and evaluate how those choices mitigate the program's software risks.

Acquisition planners can apply the method and techniques discussed in this report without using the ASDT. The ASDT is provided for acquisition planners so that each acquisition organization does not have to design templates from scratch. The ASDT is available for download on the SEI Publications Web site at
http://www.sei.cmu.edu/publications/publications.html.

Instructions on how to use the ASDT are contained within the workbook. The ASDT requires Microsoft Office Excel 2003 or newer. Macro execution must be enabled for it to work properly.

2 Developing Acquisition Strategies

2.1 Acquisition Planning Includes an Acquisition Strategy and an Acquisition Plan

Acquisition planning is defined as

> *The process by which the efforts of all personnel responsible for an acquisition are coordinated and integrated through a comprehensive plan for fulfilling the agency need in a timely manner and at a reasonable cost. It is performed throughout the life cycle and includes developing an overall acquisition strategy for managing the acquisition and a written Acquisition Plan (AP)* [DAU 03].

As shown in Figure 3, acquisition planning is the act of defining and maintaining an overall approach for a program. Acquisition planning guides all elements of program execution to transform the mission need into a fielded system that is fully supported and delivers the desired capability. The goal of acquisition planning is to provide a roadmap that will be followed to maximize the chances of successfully fielding a system that meets users' needs within cost and schedule. Acquisition planning is an iterative process; feedback loops impact future acquisition planning activities.

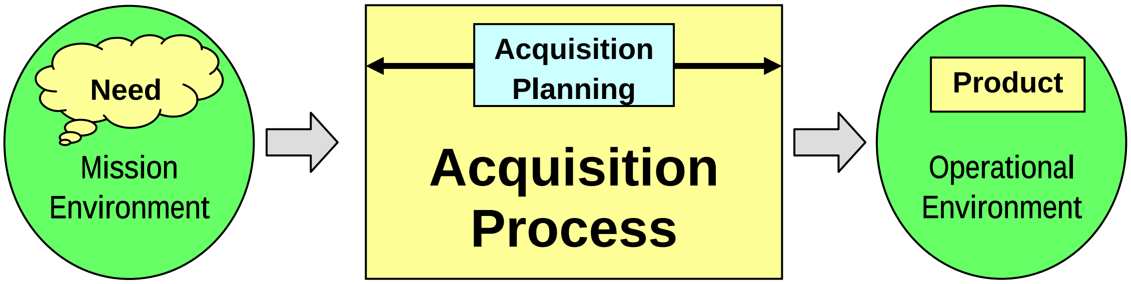


Figure 3: The Acquisition Planning Context

Software acquisition planning is a powerful process that can help define the most promising acquisition options that best mitigate risks in the development of software. Program risks take many forms and come from many sources. The first challenge to the program management team is to identify all of the affected stakeholders. The second challenge is to apply the expertise of program office staff and these stakeholders to identify and prioritize the program's goals and risks. After these two challenges are met, it is possible to reason about and create an acquisition strategy that mitigates the highest priority risks and manages the remaining risks.

An *acquisition strategy*, when formulated carefully, can be a means of addressing program risks via program structure. More formally, an acquisition strategy is

> *A business and technical management approach designed to achieve program objectives within the resource constraints imposed. It is the framework for planning, directing, contracting for, and managing a program. It provides a master schedule for research, development, test, production, fielding, modification, postproduction management, and other activities essential for program success. The acquisition strategy is the basis for formulating functional plans and strategies (e.g., Test and Evaluation Master Plan (TEMP), Acquisition Plan (AP), competition, systems engineering, etc.)* [DAU 03].

The “best” acquisition strategy for a given program directly addresses that program’s highest priority risks. High-priority risks can be technical, if no one has yet built a component that meets some critical aspect of the system or has never combined mature components in the way that is required. Risks can be programmatic if the system must be designed to accommodate predefined cost or schedule constraints. Or, risks can be mission-related when the characteristics of a system that meets the need cannot be fully articulated and agreed upon by stakeholders. Each program faces a unique set of risks, so the corresponding acquisition strategy must be unique to address them.

In addition, program risks change over the course of the program as new risks are discovered and previously identified risks are mitigated. The selected acquisition strategy must evolve to respond to what is known (and unknown) about the goals, objectives, and constraints of the system; the state of the requirements; the maturity of the technology, including the software; and the available resources such as budget, people, and skills. Much as a commander creates and adjusts a course of action based on a sense of the threat and the operational constraints and resources available to meet that threat, a program manager uses acquisition planning to continually adjust the program’s acquisition strategy to accommodate the program’s risks as they are currently understood.

Figure 4 represents a high-level graphical view of acquisition strategy evolution. First, an initial strategy is developed and executed. Then, the acquisition strategy is refined based on progress and modifying forces to the program. This cycle is repeated multiple times until the program completes.

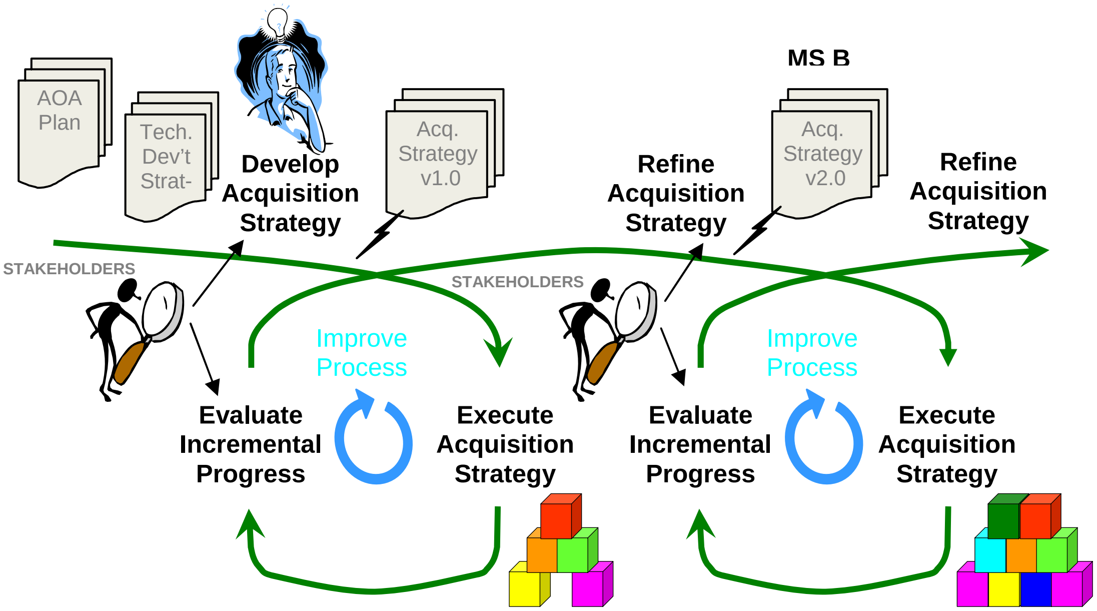

AOA Plan
Tech. Dev't Strat-
Develop Acquisition Strategy
Acq. Strategy v1.0
MS B
Refine Acquisition Strategy
Acq. Strategy v2.0
Refine Acquisition Strategy
STAKEHOLDERS
STAKEHOLDERS
Improve Process
Improve Process
Evaluate Incremental Progress
Execute Acquisition Strategy
Evaluate Incremental Progress
Execute Acquisition Strategy

Figure 4: The Acquisition Strategy Development Process

As stated previously, an acquisition strategy is one of two major components of acquisition planning. The second necessary component is a written acquisition plan. An acquisition plan is

> *a formal written document reflecting the specific actions necessary to execute the approach established in the approved acquisition strategy and guiding contractual implementation* [DAU 03]

The terms *acquisition planning*, *acquisition strategy*, and *acquisition plan* are frequently used interchangeably, which causes much confusion. A program's acquisition strategy is different from its acquisition plan, but both are artifacts of the process of acquisition planning.

2.2 Key Aspects of Acquisition Strategy Development

An acquisition strategy must take into consideration many elements to accomplish system objectives during the course of the system's life cycle [DoD 96]. An acquisition strategy includes a collection of *strategy elements*. Each strategy element is a decision or a plan on how to proceed with a specific aspect of the program execution. Examples of strategy elements include, but are not limited to, the acquisition approach, contract approach, and competition. Section 4 describes strategy elements in more detail.

The acquisition strategy guides decisions made throughout the life of the program. It should be based on reasoning about both internal and external complex factors with regard to the system being built, operated, and maintained. When developing an acquisition strategy, it is important to understand the program's driving factors. *Drivers* are conditions that impact program risk. For example, the skills of the program office personnel, the number of stakeholders, and the number of system configurations that need to be maintained are all drivers. Drivers can influence the acquisition strategy independently or there can be interactions between drivers that affect program risk. For example, unstable requirements usually influence overall costs. Section 3 contains a taxonomy to help acquisition planners profile the program and determine the extent to which it is exposed to software risk.

Figure 5 shows the relationships among the program's drivers, its risks, and its acquisition strategy. The first step is to determine a program's internal and external drivers.

After identifying the major drivers, you need to

- determine which drivers present the highest risk to your program
- determine how the drivers will influence your program's acquisition strategy elements
- formulate strategies (by making choices among alternatives) that you believe will best address the highest risk drivers
- analyze the strategies to determine gaps and remaining high risk areas and determine if these risks can be mitigated through other options

No strategy can mitigate all of a program's risk; identify the strategy that provides the best risk mitigation and can be executed by the available staff and within the available budget.

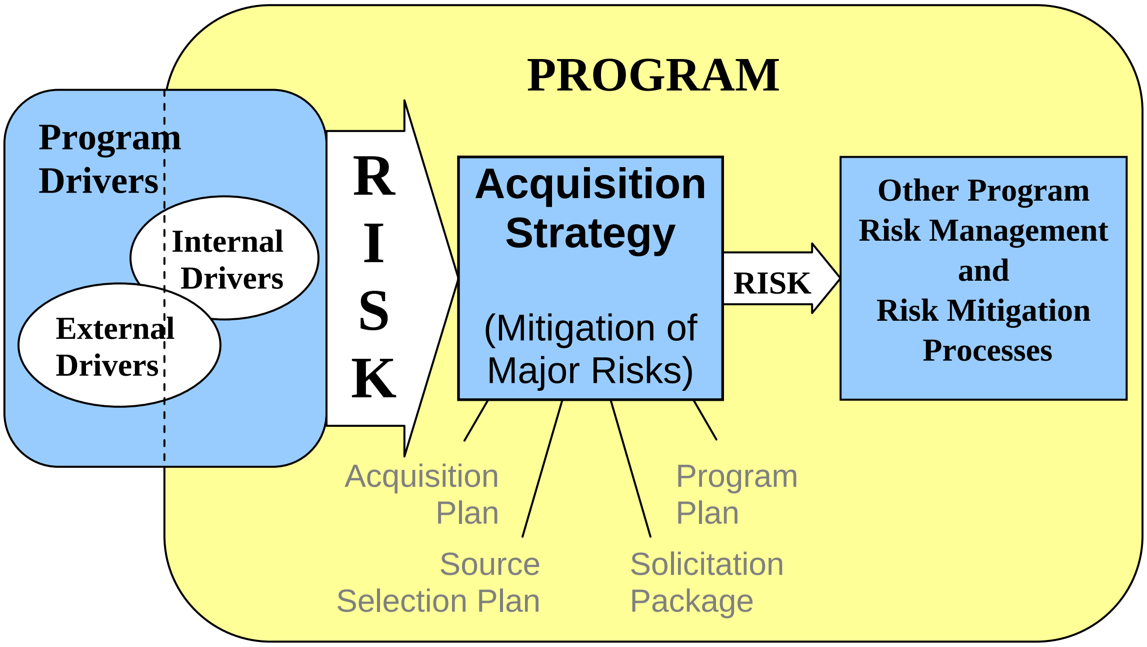


Figure 5: Risk Management Via Acquisition Strategy

2.3 Software Risk in Acquisition Planning

Acquisition planning must address the risks that can arise during development, use, and maintenance of software. To this end, the acquisition strategy must reflect

- the degree to which software determines the capabilities and qualities of each system component or sub-component
- all software elements that are potentially difficult or costly to implement
- the operational, maintenance, and sustainment implications of software components
- the threat that faulty or failed software poses to human life, safety, security, environment, and so on

Some acquisition planners neglect to address the unique nature of software risks. They ignore them or treat them as an insignificant part of the system implementation risks and assume that they will be managed by the implementation contractor. This paradigm persists for a variety of reasons.

- The acquisition community as a whole does not address software early enough in system development. Therefore, many critical acquisition planning decisions are made with scant consideration of software.
- Acquisition planners have no empirical foundation for selecting one acquisition approach over another. This makes it difficult to know whether a given approach can successfully mitigate software risks and encourages managers to fall back on the approaches with which they are most familiar; these approaches usually ignore software risks.

2.3.1 The Challenge of Software

Increasingly, software is the major determinant of system functionality. In the year 2000, the Defense Science Board found tremendous growth in the software content of both manned and unmanned systems [DSB 00]. In fact, software requirements now represent the bulk of overall specification requirements for most systems. For example, 80% of the U.S. Air Force’s requirements for building the F/A-22 Raptor and 65% of the requirements for the B-2 Spirit Stealth Bomber were software requirements [Nelson 99].

Within the U.S. Army, the growing dependence on software is equally astounding. The M1 Abrams Main Battle Tank (MBT) performs its mission with the help of approximately 600,000 (0.6 M) source lines of software code (SLOC) [Nelson 99]. Planned Army systems such as those under development by the Future Combat Systems network employ far more software (34M SLOC) [GAO 05].

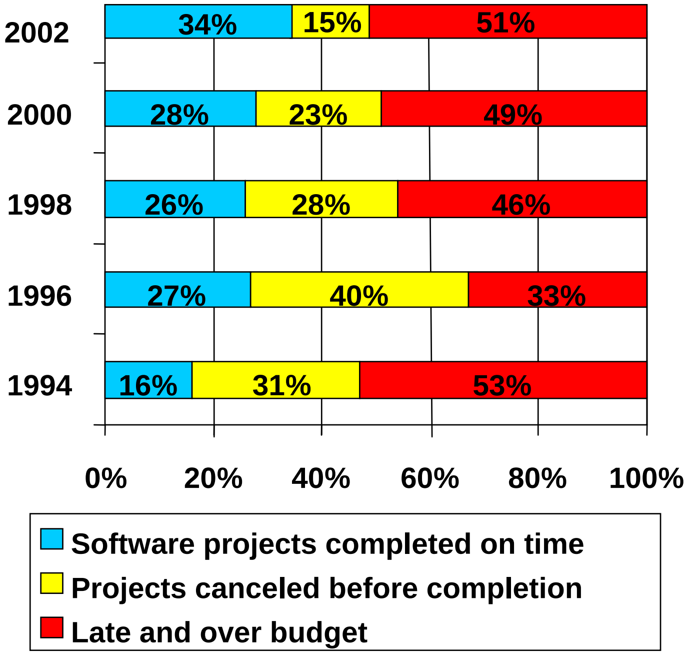


Figure 6: A Survey of Software Projects According to the Standish Group

Due to the advanced capabilities now required, software increasingly drives the cost, schedule, and quality of integrated, hybrid (software and hardware) systems. As shown in Figure 6, the percentage of software projects completed on time seems to be improving. However, in 2002, 51% of software projects were late, over budget, and/or lacked critical features—and this number excludes an additional 15% of projects that were cancelled. In addition, the average final software product contains only 52% of the originally specified features [Standish 03].

Systemic analysis of results from the Tri-Services Assessment Initiative suggest that there has been "little positive impact from past corrective actions, initiatives, and policy" in the Department of Defense (DoD) [McGarry 03]. There are many reasons why software, particularly DoD software, has proven so difficult to build. We cannot possibly list them all in this document, but we can reflect on several reasons that illustrate why formulating an acquisition strategy is critical.

1. It is likely that no organization acquires a wider range of software than the DoD. In the Army alone, software ranges from well-defined business systems to systems supporting the acquiring, tracking, and destroying of a missile moving faster than a bullet. The broad range of software required to support these two different missions implies that an equally broad range of strategies, processes, and technologies must be employed.
2. Some of the most complex types of software produced by the DoD are components embedded inside large combat systems. In many of these systems, software is an enabler that, when used in combination with other technologies, can be used to address sys-

tem/subsystem requirements. In many cases, alternative technologies could be used to provide the same functionality—albeit with different implications. The complexity of the software required is rarely evident when the system is conceived and the software requirements tend to grow in complexity as hardware component builders make assumptions about specific systems and the community of interacting systems. Eventually, software must be developed to fill the gaps and to integrate individual, custom systems.

3. Engineers in other disciplines understand the limitations of the materials and associated fabrication techniques within their discipline. For example, there are rating scales for properties of metals such as thermal stability, tensile strength, hardness, elasticity, and so on. On the other hand, there are few, if any, hard and fast "rules" of software engineering. In this environment, planners, developers, stakeholders, and users all tend to make optimistic assumptions about software—that it is infinitely malleable and can be made to exhibit unprecedented degrees of interesting characteristics (e.g., dependability, performance, and security). In effect, flexibility is the greatest asset of software but is also its key liability.

4. The DoD acquisition community is rightly focused on acquiring capabilities to support the warfighter. However, this tight focus sometimes leads to reduced emphasis on the problems of developing appropriate software to support these capabilities because the engineering of the software becomes secondary to the engineering of the hardware for the system. Because software is inherently intangible and often a relatively small part of the system, other system issues overwhelm it. As a result, software development is not properly planned, started late, defined late, under-budgeted, and so on. Ultimately, a software implementation that might have been straightforward at one time stagnates until there is no option but to attempt to develop a now-complex software component on a shortened schedule.

Effective software acquisition can be a point of great leverage. Addressing even some of the problems with software in major system acquisitions could lead to huge savings and better systems. The DoD has recognized this for many years and has attempted to improve the situation by instituting various standards such as DoD-STD-2167A, MIL-STD-498, IEEE/EIA 1207, by embracing process improvement methodologies such as ISO 9000 and the Capability Maturity Model® Integration (CMMI®) framework, and by adopting a wide range of newer, more flexible technologies such as Extensible Markup Language (XML). However, in almost all cases, as evidenced by the Tri-Services Assessment Initiative, results have been disappointing [McGarry 03].

2.3.2 Planning to Mitigate Software Risk

Software acquisition planning is a powerful process that can help define the most promising acquisition options that best mitigate risks in the development of software. An acquisition strategy should clearly describe the program's approach for defining, building, fielding, and supporting the system by

- documenting what the program understands to be the need and the environmental constraints under which the system must be built, operated, and maintained
- defining the means by which requirements that reflect the need will be developed and managed
- establishing appropriate relationships and commitments among stakeholders, program offices, and contractors for the life of the system
- providing a framework under which a contract (or contracts) is managed to deliver products or services needed by the program
- establishing mechanisms to effectively communicate the approach that is used by the program to all affected stakeholders

A software acquisition strategy must effectively communicate a complete, coherent, and consistent approach that is used by the program so that the program can be resourced, staffed, and tracked. Above all, the software acquisition strategy must support the DOD directive to

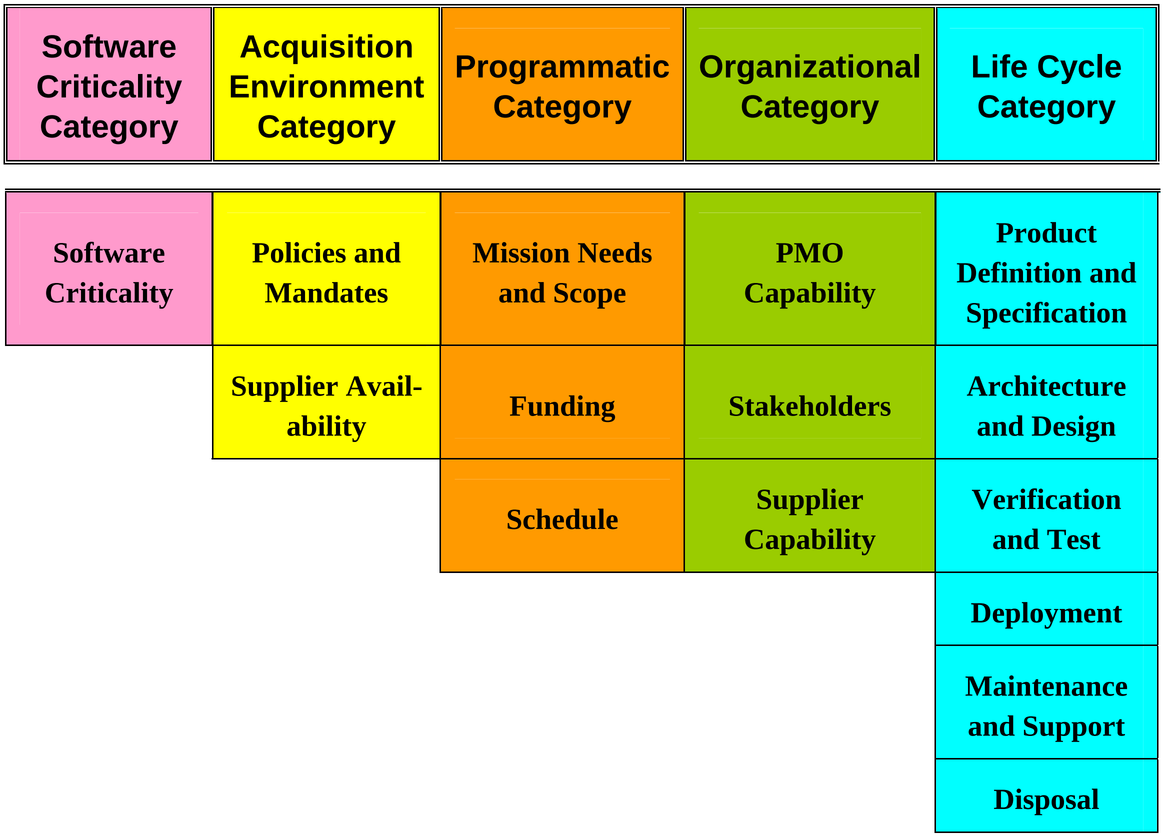
> *acquire quality products that satisfy user needs with measurable improvements to mission capability and operational support, in a timely manner, and at a fair and reasonable price* [DoD 03a]

Software acquisition planning aimed at mitigating software risk begins as soon as it becomes apparent that a materiel solution employing software might be required—or in the earliest days of Concept Refinement for a program that uses all of the acquisition phases described in *DoD Instruction Operation of the Defense Acquisition System (DoDI 5000.2)* [DoD 03b]. At no other point is the program as pliable. A software acquisition strategy is custom-built to help the system stakeholders, program management office[1] (PMO), and contractors achieve the best software system possible within program constraints. While these early steps in acquisition planning are the first and arguably the foremost mechanism available to achieve objectives in the acquisition and sustainment of systems, the acquisition strategy must continue to evolve as concepts are refined, as technology is developed, as systems are acquired, and as systems are fielded and supported—until the systems are retired.

[1] The term *program management office* (PMO) is used primarily by civil agencies and the U.S. Army to refer to an organizational unit created to centralize and coordinate the management of projects under its domain. Note that different government organizations use different terms to refer to this organizational unit. For example, the Air Force uses the term “system program office,” or “SPO,” while the Navy refers to it as a “project management office. In the interest of simplicity and readability, the authors of this report use the term “PMO” in a general way.

3 Profiling Program Software Characteristics and Acquisition Strategy Drivers

This section introduces a taxonomy of categories and drivers, depicted in Figure 7, to help a program determine the extent to which it is exposed to software risk. The categories and corresponding drivers are designed to be used by acquisition planners to profile software risk early in the program. The software risk profile should be updated throughout the execution of the program and the acquisition strategy should be adjusted to address newly discovered risks and risks that have been mitigated.



Software Criticality Category	Acquisition Environment Category	Programmatic Category	Organizational Category	Life Cycle Category
Software Criticality	Policies and Mandates	Mission Needs and Scope	PMO Capability	Product Definition and Specification
	Supplier Availability	Funding	Stakeholders	Architecture and Design
		Schedule	Supplier Capability	Verification and Test
				Deployment
				Maintenance and Support
				Disposal

Figure 7: Categories of Program Software Characteristics and Subsequent Strategy Drivers

As shown in Figure 7, program characteristics and strategy drivers are aggregated into five main categories:

1. *Software criticality* quantifies the extent to which the system is "software-bound." Systems are considered *software-bound* when they require large amounts of software, highly complex software, or software that is directly responsible for fulfilling critical aspects of the system mission.
2. *Acquisition environment* program characteristics and associated strategy drivers quantify the extent to which the environment in which the acquisition occurs affects the software risk in the program. The level of acquisition environment risk can be represented in terms of policies and mandates imposed on the program and the availability of suppliers.
3. *Programmatic* program characteristics and associated strategy drivers quantify the extent to which the fundamental programmatic aspects of scope, funding, and schedule affect the software risk in the program.
4. *Organizational* program characteristics and associated strategy drivers quantify the extent to which organizational factors inside and outside the program affect the software risk in the program. The level of organizational software risk can be represented in terms of the characteristics of the program management office, supplier, and stakeholders.
5. *Life-cycle* program characteristics and associated strategy drivers quantify the extent to which various life cycle facets pose risk to the program. The level of life-cycle software risk can be represented in terms of the risks associated with the definition, specification, architecture, design, implementation, test, deployment, operations and maintenance, and disposal of the system.

Each of these categories, subsequent strategy drivers, and sublevel drivers is discussed in more detail in the following sections. One or more of the drivers may be difficult to determine or not applicable during a particular program time frame.

3.1 Software Criticality Category

3.1.1 Software Criticality Driver

The extent to which an individual program must specifically focus on and address software risks depends on the criticality of software within the capability to be deployed. To determine software criticality proportion and reliance need to be evaluated.

Proportion is the ratio of the software to the entire system and is usually evaluated using life-cycle cost, time needed to produce and maintain the software, and functionality provided. For example, is software the majority of the system or a minor part of the system? For certain categories of business systems, such as financial, human resources, inventory management systems, and for other categories of systems that support the mission of the DoD such as Command, Control, Communications, Computers, Intelligence and Reconnaissance systems, the proportion of the software to the system is obvious. For these types of systems, software virtually is the system.

For other types of systems, however, the significance of software components is not as obvious. In these systems, the software may not be configured as a set of large applications that appear on a software architecture diagram at the system level, but instead, software exists in the system as relatively small chunks distributed across a number of other systems or devices. In this case, the relationship between the software components and hardware components may be more dominant than the relationship between the individual pieces of software. The proportion of software might be small, but the degree to which the important functionality of the system depends on software is large. That is why an evaluation of reliance is necessary to determine software criticality.

Reliance is the degree to which the important functionality of the system depends on the software. For example, a system running flight control software that is needed to fly a helicopter has a *higher reliance* on software than a system that employs software as a means of modeling certain system characteristics.

What can be deceiving about the criticality of the software to be procured, however, is that in many modern systems, evaluating proportion alone is insufficient and the reliance the system has on the software must also be evaluated. For example, modern cars—like their 1909 Model T counterparts—are systems that rely on a drive train to impel the wheels, a steering linkage to control the direction of travel, and brakes to stop them. Unlike the Model T, however, virtually no modern car can perform in more than a “limp home” mode without software to control many mechanical functions.

The software used by the 2003 BMW 745Li, shown in Figure 8, is a good illustration of the increasing software control of hardware systems in automobiles. Among the many features that are likely to rely on software are the “mayday” phone, on-board navigation system, dynamic stability control, dynamic traction control, active roll stabilization, dynamic brake control, coded drive-away protection, an adaptive automatic transmission, and iDrive systems. This list can be extended to include the software that controls engine performance, emissions, and fuel consumption found on virtually all modern cars.

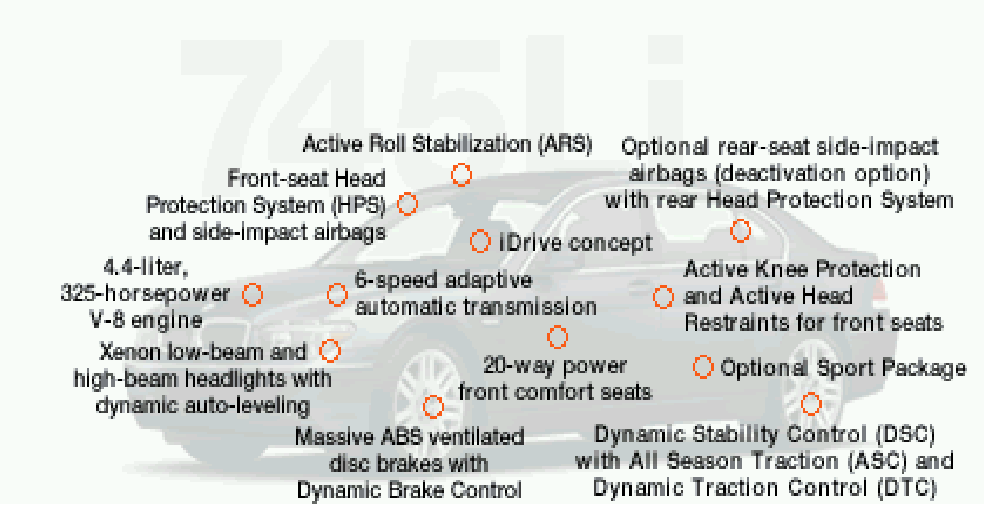

745Li
Active Roll Stabilization (ARS)
Optional rear-seat side-impact airbags (deactivation option) with rear Head Protection System
Front-seat Head Protection System (HPS) and side-impact airbags
iDrive concept
4.4-liter, 325-horsepower V-8 engine
6-speed adaptive automatic transmission
Active Knee Protection and Active Head Restraints for front seats
Xenon low-beam and high-beam headlights with dynamic auto-leveling
20-way power front comfort seats
Optional Sport Package
Massive ABS ventilated disc brakes with Dynamic Brake Control
Dynamic Stability Control (DSC) with All Season Traction (ASC) and Dynamic Traction Control (DTC)

Figure 8: BMW 745Li Software

The trend toward increased software control can also be seen in U.S. military systems. For example, in the 1975 version of the F-15 Eagle only 35% of functions required software support. That number grew to 80% for the F/A-22 Raptor introduced 20 years later [Nelson 99].

One definition of the criticality of the software in a system can be found in Table 1.

Table 1: Relationships Among Criticality, Reliance, and Proportion

Criticality	Reliance	Proportion
Very High	Complete reliance on software	Majority of the system
High	High reliance on software	Majority of the System
Medium	High reliance on software	Minor part of the system
Low	Low reliance on software	Minor part of the system
Very Low	No reliance on software	Software is not part of the system

The software in a system (or software used to model, simulate, or test a system) cannot be categorized simply as critical or not. Most programs categorize the software somewhere between the two extremes. The extent of the software criticality can be rated on a sliding scale that extends from one extreme to the other: Very Low to Very High.

If software criticality is low, software risks are not dominant factors in acquisition planning. However, systems that are not software-critical are increasingly rare. Therefore, most programs today need a way to reason about software risks and the potential for the chosen acquisition strategy to mitigate software risk within the system.

3.2 Acquisition Environment Category

Evaluating acquisition environment program characteristics and associated strategy drivers, including policies and mandates and supplier availability, can help quantify the type and amount of software risk in the program.

3.2.1 Policies and Mandates Driver

Policies and mandates are external constraints that a higher authority imposes on a program. One definition of *policy* is "a definite course or method of action selected from among alternatives and in light of given conditions to guide and determine present and future decisions" and another is "a high-level overall plan embracing the general goals and acceptable procedures especially of a governmental body." The dictionary also defines a mandate as "an authoritative command" that a higher authority imposes [Merriam-Webster 05].

Policies and mandates come in several forms and can be described by the examples shown in Table 2.

Table 2: Examples of Policies and Mandates

Policy or Mandate	**Example**
Compliance with one or more laws	• OSHA • Privacy Act
Compliance with one or more directives or instructions	• Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) • Business Enterprise Architecture (BEA) • Joint Technical Architecture (JTA) • DoD 8500 series on Information Assurance
Compliance with a specific practice	• Use of performance requirements instead of detailed specifications • Use of a Statement of Objective (SOO) instead of a Statement of Work (SOW) in the initial solicitation • Use of a Capabilities Development Document (CDD) instead of an Operational Requirements Document (ORD)
Compliance with specific contracting elements	• 8A requirements
Required certifications and accreditations	• Certificate of Networthiness (CoN) • Certificate to Operation (CtO)
Use of Government Furnished Information (GFI) and Equipment (GFE)	• Incorporation of a specific component • Providing the contractor with a development and testing environment

When formulating an acquisition strategy, acquisition planners must consider the amount of conflict between the policies and mandates imposed on a program, the relative stability or instability of a particular policy or mandate, the degree to which policies and mandates conflict with program needs, and the amount of control a program has to modify a policy or mandate.

Analyzing the applicability of mandates, resolving any conflicts, and determining their impact, regardless of their relevance to the program, requires resources. Traversing the paper trail to verify compliance with policies and mandates can be quite time-consuming; these tasks should not be underestimated during planning. Information about regulations abounds, but gathering information about every single policy and mandate that applies to a program is a nontrivial task. For example

- Who is responsible for figuring out the detail of compliance: the PMO or the contractor?
- What is the risk something will be overlooked?
- What is the risk that a policy or mandate can be interpreted in multiple ways?

The more policies and mandates imposed on a program, the more resources consumed, and more likely that one policy or mandate will conflict with another policy or mandate. Also, after a conflict between mandates is identified, there may or may not be a consistent mechanism to resolve the conflict and programs confronted with such conflicts rarely receive guidance on how to resolve them. The PMO may assume some risk when the contractor resolves a conflict that the PMO doesn’t know about or if the PMO does not fully understand what the resolution means. Whether the PMO or contractor attempts to resolve a conflict between mandates, there is a risk that the resolution can cause technical incompatibilities with other systems in the future or inefficiencies in the current system.

In addition to conflicts arising between individual policies and mandates, conflicts between the policies and mandates and the needs of the program can also arise. These types of conflicts can be manifested in many forms including

- expense of conforming to policies and mandates without having additional funding to do so. For example, a requirement to use a specific software component might be imposed, but funding for the sustainment of that component is not provided or guaranteed.
- requirements that specific practices be followed (or artifacts created) that are untried and unproven and have not been adequately piloted before being imposed. For example, one program producing real-time embedded software was directed to develop all code in Java at a time when Java did not support real-time software.
- expense of significant rework and schedule degradation when policies or mandates are levied changes after the system has already been designed
- requirements to use a solution developed by, or for, the DoD that competes with more viable commercial alternatives

- schedule disruption caused by mandates. For example, meeting National Security Agency certification requirements can become a critical path in the program schedule and must be planned.
- work to determine which policies and mandates apply to your program and what the impacts are; in a large program, determining how the policies and mandates can be applied consistently to all facets of the program. Even allocating the proper staff to monitor these issues can be difficult for a PMO.

As in the case of conflicts between policies and mandates, after a conflict is identified between policies and mandates and program needs, there may or may not be a consistent mechanism to resolve the conflict. In addition, alternative solutions may not be allowed.

The stability or instability of a policy or mandate must also be considered. Is there a risk that the program will adhere to a policy or mandate that will change after key program decisions have been made? What is the likelihood that a policy or mandate will be introduced midway through the program that would require the solution to be reworked?

A program's ability to resolve policy and mandate conflicts may counterbalance the risk imposed by the number and extent of the conflicts. It is important to have a good understanding of which conflicts are beyond PMO resolution so that appropriate higher level help can be sought. The presence of too many irresolvable conflicts can be a warning sign that the program cannot be executed as currently planned.

The extent to which a program has policy and mandate conflicts can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.2.2 Supplier Availability Driver

Another external environmental factor to consider when evaluating an acquisition strategy is the number of available, qualified suppliers of required software components. How many suppliers can potentially fulfill the needs of the program? Is the program constrained by a limited set of pre-qualified suppliers? Does their expertise match the needs of the program? Are there unique aspects of the program that can only be supported by a limited number of suppliers? Do the suppliers have adequate capacity to execute all of the expected work within the required time frame? The number of available, qualified suppliers can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.3 Programmatic Category

Programmatic program characteristics quantify the extent to which fundamental programmatic strategy drivers of mission needs and scope, funding, and schedule affect the type and amount of software risk in the program.

3.3.1 Mission Needs and Scope Driver

Systems are developed to fulfill the needs of the mission. Every acquisition plans to fulfill all defined mission needs and deliver a product that satisfies 100% of the user requirements. And yet we see from the Standish Group's CHAOS report that "successful" systems meet only 52% of requirements [Standish 03]. Clearly, the mission needs contain some degree of flexibility. The flexibility of the mission that the system needs to fulfill is a strategy driver. For some missions, the scope of the mission is very exact and constricted. For others, it is much more flexible. The extent to which the mission needs are constrained can be rated on a sliding scale that extends from one extreme to the other: Flexible to Rigid.

3.3.2 Funding Drivers

The funding drivers described in this section include funding constraints and the funding profile.

3.3.2.1 Funding Constraints

Funding constraints determine the degree to which the amount of funding allocated to the program matches the estimated funding required to develop, operate, and maintain the system. Funding constraints can significantly impact both "software-critical" and "non-software-critical" systems. Inadequate funding can cause suboptimal engineering choices to be made in the interest of saving money; choices such as deferring functionality or allocating requirements to software that might be better implemented in hardware.

How does software influence the risk associated with funding constraints? A key consideration is the life-cycle cost and total cost of ownership estimates for the software in the system. According to DoDI 5000.2, sound cost estimates are based on well-defined programs. With large-scale unprecedented systems, this becomes a difficult task.

The inability to develop credible and accurate cost estimates correlates with the level of uncertainty in the program or system definition, technical performance, and cost estimating method. In turn, this influences the accuracy of the anticipated funding profile. In contrast, the funding profile is sometimes dictated by the congressional budget. The program must then use its cost estimating ability to determine an affordable set of requirements.

Factors influencing cost estimating include immature software cost estimation models, lack of experience in using cost models, lack of knowledge and experience in planning and managing software development programs, lack of subject matter experts, immature and volatile requirements, the quality of historical data from similar projects, and influences by executives having the authority to reject or modify valid estimates due to "political" or other reasons.

The extent to which the funding constraints are a fact of the acquisition can be rated on a sliding scale that extends from one extreme to the other: Few to Many.

3.3.2.2 Funding Profile

The type and timing of funding dollars applied to the acquisition can also drive decisions made about the acquisition strategy. It is not just how much funding the program receives, but also the type of appropriation and when it is received. For example, a program that needs to do a lot for research and development and will acquire a limited amount of hardware usually has a front-loaded budget, in which a majority of the overall funding is allocated to do the work at the beginning of the project versus the end. In addition, this program would need research and development dollars.

In order to develop high levels of product quality, investment in up-front engineering tasks is necessary. Tasks such as requirements definition and architecture drive the scope of the software development effort, so inadequate attention to those tasks invariably leads to increases in software size and complexity and ultimately to schedule delays, cost increases, and product quality issues. Consequently, early engineering tasks need to be funded adequately at the outset of the program.

Using an evolutionary acquisition approach such as incremental or spiral can complicate funding needs. For example, the program may need to budget funding from several different appropriation types simultaneously.

The extent to which the funding profile matches the needs of the acquisition can be rated on a sliding scale that extends from one extreme to the other: Mismatched to Matched.

3.3.3 Schedule Drivers

The schedule drivers described in this section include schedule constraints and urgency.

3.3.3.1 Schedule Constraints

Like funding constraints, schedule constraints are the degree to which the schedule allocated to the program matches the estimated time required to develop, operate, and maintain the system. Schedule constraints can significantly impact both “software-critical” and “non-software-critical” systems.

How does software influence the schedule for the system? What aspects of the program have key dependencies on certain software events? If software is on the critical path, software risks need to have very good mitigation plans and must be constantly monitored.

The ability to develop credible and accurate schedule estimates is influenced by the level of uncertainty in the program or system definition and the schedule estimating method used. Other factors that influence schedule estimating include immature estimation models, lack of experience in using the models, lack of knowledge and experience in planning and managing software development programs, lack of subject matter experts, immature and volatile requirements, the experience of the personnel performing the estimates and the quality of historical information from similar projects, and influences by executives having the authority to reject or modify valid estimates due to "political" or other reasons.

Another consideration that can have a large impact on schedule risk is the need to interface with other programs. Risk is increased if, due to interoperability goals, the program schedule must be coordinated with one or more other programs.

The extent to which schedule constraints are a fact of the acquisition can be represented as a continuum that extends from one extreme to the other: Few to Many.

3.3.3.2 Urgency

In certain circumstances, urgency of deploying a capability to the field might have significant impact on the acquisition strategy. For example, the current war situation might be placing pressure on certain acquisitions to deliver capabilities at a different rate, in a different sequence, or with different capabilities than planned. How does your acquisition strategy accommodate urgent needs? The level of urgency can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.4 Organizational Category

Organizational program characteristics and associated strategy drivers quantify the extent to which organizational factors inside and outside the program affect the software risk in the program. Strategy drivers include PMO capabilities, stakeholders, policies/mandates, and performance team/contractor capabilities.

3.4.1 PMO Capabilities Drivers

The drivers pertaining to PMO capabilities described in this section include staff skills, capacity and stability, and the process focus of the PMO and its governing organization.

3.4.1.1 PMO Staff Skills

> *The PM* [program manager] *must assemble the proper acquisition strategy selection and development team. It is important to staff this team with individuals whose knowledge, experience and access to pertinent information equips them to effectively address all of the topics. The success of each of the succeeding steps in the selection process depends on the active participation of all the members of the team. Good contracting, technical and business/financial managers will be key payers in the selection of the acquisition strategy* [Department of the Navy 01].

With respect to having the proper software expertise, the GAO noted: "The more managers know about software development processes and metrics, the better equipped they are to acquire software” [GAO 04a]. In effect, the government program manager needs to be a “smart buyer” of the software within the system.

While some level of software expertise is necessary, it is not the only ingredient in the recipe for success. Systems engineering is also a critical skill and an inherent PMO responsibility. Therefore, the PMO must also have adequate systems engineering staff. In addition, for software critical systems expertise is needed in all areas including software management (cost and schedule estimating, metrics definition and analysis, process improvement), software acquisition, software architecture, software and systems integration, software development (including commercial off-the-shelf [COTS], reuse, and open systems), security and accreditation and organizational change management.

The expertise needs to be brought on early in the program to develop the acquisition strategy, support RFP preparations, review proposals, contribute to program development, and provide program oversight. The staff needs to maintain their skills and be familiar with the latest techniques over the course of the acquisition. A gap analysis should be performed periodically to identify any gaps and a corresponding training, rotation and development plans put in place to address the missing skills.

The extent to which the PMO has the appropriate level of skills it needs to execute the acquisition can be rated on a sliding scale that extends from one extreme to the other: Weak to Strong.

3.4.1.2 PMO Staff Capacity

In addition to having the appropriate skills as described in Section 3.4.1.1, the PMO also has to have the right number of people with those skills to plan and execute the acquisition. It is important to be able estimate how many people you need, with what skills, and at what time. For example, you may not need three or four test experts while you are preparing the acquisition strategy, but you may need that many to oversee the test phase. What skills does the program need when? How much control over staff capacity and the escalation process for resolving staffing issues that are out of your control does the program have?

The degree to which the staff members have worked together before must also be factored into the schedule. Are they already a team or do they have to go through the required team-building steps before they function optimally?

The extent to which the PMO has the staff capacity it needs to execute the acquisition can be rated on a sliding scale that extends from one extreme to the other: Inadequate to Adequate.

3.4.1.3 PMO Staff Stability

Permanent Change of Station (PCS) and Permanent Change of Assignment (PCA) are facts of life in the military. The program must take this into account when formulating its acquisition strategy. How does the acquisition strategy compensate for the fact that the average time an active duty person spends in a PMO is roughly 2–3 years? Or, that the higher the rank, the more likely a person will be reassigned? Programs also have to evaluate the stability of civilian employees or contractors supporting the PMO.

How will continuity and history be maintained for the program? What will prevent a new software architect from completely changing course? What will help a new integration and test lead do his or her job quickly and effectively without compromising the quality of the system? Planning for continuity in the event of personnel reassignment or general staff reduction is an essential part of acquisition planning.

The extent to which the PMO staff is stable can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.4.1.4 PMO Process Focus

A process focus is just as important to the acquisition organization as it is to the suppliers with which the acquisition organization works. Are there solid, repeatable processes or are things done in an ad hoc fashion? Is there a focus on improving their processes through the course of the acquisition? How do the PMO processes mesh with the supplier processes? The extent to which the PMO understands and uses process and process improvement initiatives during the acquisition can be rated on a sliding scale that extends from one extreme to the other: Weak to Strong.

3.4.2 Stakeholders Drivers

Stakeholders are individuals or groups with vested interest in the system. For example, end users, system integrators, acquirers, and sponsors are all stakeholders. Each of these stakeholders' concerns can add unique risks, many of which are discussed in other sections. For example, how willing are users able to adapt and learn a new system? How much experience does your integrator have with this type of system? How well staffed is the acquisition organization? How well do the sponsors communicate their requirements?

The stakeholders drivers described in this section include the number and diversity of stakeholders, the level of stakeholder engagement, and the level of stakeholder agreement.

3.4.2.1 Number and Diversity

The sheer number and diversity of stakeholders in the acquisition impacts the acquisition strategy: the more stakeholders the PMO has to interface with, negotiate with, and keep informed, the higher the risk for the acquisition. It is essential that the PMO have a good understanding of how many stakeholders are involved in the program. A complex set of stakeholders increases software risk for the following reasons.

- There is increased probability of divergent expectations and requirements (see Section 3.5.1).

- Tradeoffs will not have equal impact on all parties—there will be winners and losers.
- Communication becomes more complex
- Coordination takes more time; there will be multiple approval paths and associated bottlenecks.
- The amount of education required to discuss software issues increases.

Acquirers must be careful to include all the relevant stakeholders, since omissions can affect drivers and risks. This does not imply, however, that all stakeholders should necessarily have an equal voice in each phase of the acquisition.

The number and diversity of stakeholders can be rated on a sliding scale that extends from one extreme to the other: Small to Large.

3.4.2.2 Level of Engagement

The level of engagement by stakeholders in the acquisition process can affect the acquisition strategy. One way to describe the level of engagement is by monitoring the scope of stakeholders' involvement, the quality of their input and feedback, and their responsiveness. In the worst case scenario, you would get lots of input in an untimely fashion that is of poor quality with a narrow view of the system. In the best case scenario, you would get lots of input in a timely fashion that is of high quality from a broad view of the system. How will the program ensure proper representation, opportunities for participation, and commitment to deadlines from stakeholders? The level of engagement can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.4.2.3 Level of Agreement

Stakeholder agreement is the amount of consensus among the stakeholders during the acquisition. Stakeholders should be proactively included in all phases of the acquisition to maximize buy-in and system acceptance. The impact of any change must be considered from all stakeholder perspectives. If you don't involve them, you won't gain commitment, which usually results in dissatisfaction with the system in some form, may cause delays in the program, and so on. Even if stakeholders are proactively included, there might be significant disagreement on the requirements for the system, the priority of the requirements, on the schedule, on the deployment plan, and so on. How does the acquisition strategy handle this?

Another aspect to consider is stakeholder turnover. Military stakeholders suffer from the same PCS and PCA factors as the PMO. Sometimes a change in a key stakeholder can force the PMO to revalidate decisions that were made earlier in the program.

As agreement decreases, risk increases for the program. The level of agreement can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.4.3 Supplier Capability Drivers

Acquisition planning does not stop after a contract award is made; it is a continuous process. Many programs fail to evaluate drivers that play an important role after the program is in execution. A category of drivers that is often not considered is the capability of the supplier over time.

The supplier is the team that is responsible for producing the system and its associated artifacts. Typically, in large government contracts this is usually one or more contractors, but that is not always the case. The supplier could be another government entity. In this section, we refer to suppliers as all parties responsible for supplying the system to the acquirers.

The drivers pertaining to supplier capabilities described in this section include supplier staff skills, capacity and stability, and performance to date.

3.4.3.1 Supplier Staff Skills

A supplier is awarded a contract based on its ability to achieve the objectives associated with a statement of work. For software-critical systems, expertise may be needed in all of the areas described in Section 3.4.1.1. Other expertise may also be needed, depending on the domain.

During program execution, an evaluation of the supplier skill set is critical so that the acquisition strategy can be adjusted to mitigate any potential risks in this area. For example, if the supplier lacks testing skills, consider hiring an independent test agency.

The extent to which the supplier has the skills needed to execute its responsibilities on the program can be rated on a sliding scale that extends from one extreme to the other: Weak to Strong.

3.4.3.2 Supplier Staff Capacity

In addition to having the right skills, the supplier also has to have the right number of people with those skills at the right time to meet its commitments on the program. The extent to which the supplier has the staff capacity it needs to meet its commitments can be rated on a sliding scale that extends from one extreme to the other: Inadequate to Adequate.

3.4.3.3 Supplier Staff Stability

Usually, the supplier does not have the same issues with PCSs and PCAs as the PMO. It does, however, have to manage the external pressure of the job market in order to retain key people. Software engineers, system engineers, and architects are usually highly sought after resources. The hotter the job market in certain locations, the more difficult it may be for the supplier to retain its personnel.

Other factors that cause attrition include promotions within the supplier organization, transferring to the latest “hot” program, poor program management, low morale, and significant overtime.

The extent to which the supplier’s staff is stable can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.4.3.4 Supplier Performance to Date

As stated in Section 3.4.3.1, a supplier is awarded a contract based on its ability to achieve the objectives associated with a statement of work. Typically, some form of past performance criteria are used during the evaluation process. During contract execution, the PMO needs to ensure the performance of the supplier does not add additional risk to the program. As the program progresses, the acquisition strategy can be modified as performance weaknesses or strengths are identified.

The extent to which the performance of the supplier to date can be rated on a sliding scale that extends from one extreme to the other: Poor to Excellent.

3.5 Life Cycle Category

Life cycle-related program characteristics and associated strategy drivers quantify the extent to which characteristics of various life-cycle phases affect the software risk in the program. Life-cycle strategy drivers include product definition and specification, architecture and design, verification and test, deployment, operations and maintenance, and disposal.

3.5.1 Product Definition and Specification Drivers

> *A great deal of management attention is placed on the requirements setting phase because missing, vague or changing requirements tend to be a major cause of poor software development outcomes* [GAO 04a].

The product definition and specification drivers described in this section include the volatility and understanding of the requirements, quality attribute definitions, and interoperability.

3.5.1.1 Requirements Volatility

Stable, fully defined, unambiguous, consistent, complete, and testable software requirements are rare. Having flexible requirements is not a bad thing—trying to fully define software requirements too early or trying to limit requirements changes in a changing environment may be riskier than allowing some level of flux. The acquisition strategy needs to accommodate the degree to which requirements can or should change and how that change is to be managed. For example, the acquisition could be broken into several phases to ensure you are ready to move forward and are ready to negotiate or compete the follow-on phase.

Requirements changes can come from multiple sources including new or evolving mission needs, new or evolving technology (refresh and obsolescence), modifications to budgets, new or evolving policies or mandates, and changes in the understanding of the users over time. The program needs to understand how volatile or how stable the requirements for the system (and software) are at any given time in the life cycle. The amount of requirements volatility has a significant impact on the acquisition strategy for the program. Programs seldom have requirements that are all stable or all unstable—some requirements are firm from the start, some cannot be defined until other things about the system are known, some requirements may be in a constant state of flux as technology, COTS products, and mission needs (or the understanding of what is needed) evolve. There may be problems identifying “all” of the stakeholders, or after they’re identified, getting them involved with the tradeoff and decision process. What one stakeholder wants may contradict what another stakeholder believes to be the correct product. The implication is that the acquisition strategy may have to have characteristics of either extremes or compromise between them.

How volatile are the system requirements? How volatile are the software requirements? How volatile are the hardware requirements? If hardware requirements are stable but the software requirements are not stable this driver can be split into hardware requirements volatility and software requirements volatility since different acquisition strategies could potentially be applied to the hardware and software components of the acquisition. The same can be said for system-level requirements and hardware/software requirements. Over time, the volatility of the requirements should decrease and modifications to the acquisition strategy can be made. If the volatility of the requirements does not decrease as expected, this is a good indicator that scope is not well-defined and that there is requirements creep.

The extent to which a program’s requirements are volatile can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.1.2 Requirements Understanding

As stated in Section 3.5.1.1, fully defined, unambiguous, consistent, complete, and testable software requirements are rare. Some common issues with the understanding of software requirements include

- Requirements are specified at too high of a level and there is no mechanism to refine them. Vague requirements are subject to multiple interpretations.
- Ineffective techniques are used to solicit the requirements from the appropriate stakeholders.
- The operational concept is not sufficiently detailed. For example, the operational concept might only address a subset of the users; it might be too sparse, which leads to misinterpretation, or it might contain conflicting requirements.
- Users try to extend the functionality of the system past the “need to have” state into the “want to have” state.
- Interfaces are not well understood, different than documented, or not specified at all.

- Stakeholders are not involved in the process or they are involved and the importance of their input is ignored or minimized.
- Requirements are not measurable, with clear acceptance criteria.
- A common vocabulary is not shared or there is not the same level of domain expertise between the users, acquisition team, and performance team/supplier. For example, the requirement "Display a negative number" might be interpreted by a developer as "use a minus sign" when in reality, the user is an accountant and is expecting parentheses to be used.

The extent to which a program's requirements are understood by the users, acquisition organization, and performance team/supplier can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.1.3 Quality Attribute Definition Quality

Quality attributes, or non-functional requirements, are vitally important in ensuring appropriate software architecture. Non-functional requirements are used to express some of the "attributes of the system" or "the system environment" [Leffingwell 03]. They are usually called the "-ilities," quality attributes, or general requirements. Examples include reliability, scalability, and maintainability. Not all quality attributes are "-ilities"; performance and security are two non-functional requirements or quality attributes. For example

- How dependable does the system need to be?
- What level of security does the software have to provide?
- What availability is needed? How will this be balanced with needed maintenance time frames? Is one hour a month downtime for maintenance tolerable?

Since non-functional requirements are requirements, Sections 3.5.1.1 and 3.5.2.2 apply to this type of requirement. It is particularly common for non-functional requirements to be defined without measurable, clear acceptance criteria. For example, "the page loads quickly" is inadequate. How fast must the page load under what conditions on the system?

Too often, requirements are thoroughly defined for the system functionality but not for the quality attributes. Failure to define the attributes the system needs to fulfill puts the program at great risk. How do you know which quality attributes are important for your program? How will you verify that your program has addressed its quality attribute requirements adequately? The extent to which a program's quality attributes are defined and understood can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.1.4 Interoperability

Interoperability is the ability of a collection of communicating entities to (a) share specified information and (b) operate on that information according to an agreed operational semantics. It is a multi-dimensional aspect of system engineering. The scope is far greater than simply interoperability of data. It includes, among other things, the degree of coupling and ownership. It also includes interoperability at the organizational and management levels [Brownsword 04].

One model of interoperability examines three different interoperability dimensions:

- Programmatic – Activities performed to manage the acquisition of a system. The focus is on the contracts, incentives, and practices such as risk management. It is very important to think about how your program will interface to other programs. Will there be any impact on your schedule due to delays on a program building a system/service you need?
- Constructive – Activities performed to create and sustain a system. The focus is on architecture, standards, and COTS. How many interfaces are software-related and how difficult are they to implement? How will you govern and control the interfaces? How will testing be performed? Does the information mean the same thing in my system as in the other system(s)? Which is the best system service to use?
- Operational – Activities performed to operate a system. The focus is on interactions with other systems (programs) and with users. How will the concept of operations be managed? How will consensus be reached on priorities? What is the appropriate mix of capabilities for your system? [Morris 04]

The number of interfaces and their characteristics can be rated on a sliding scale that extends from one extreme to the other: Simple to Complex.

3.5.2 Architecture and Design Drivers

The software architecture plays a key role in the overall system design because it

- embodies the earliest software design decisions that have the greatest impact on the system and may be difficult to specify early in the program
- largely determines a system’s ability to achieve its desired qualities (i.e., non-functional requirements) such as performance, security, reliability, modifiability, and others
- enables or inhibits systematic software reuse and COTS integration
- provides a means to analyze and predict a system’s behavior
- enables the supplier to subdivide work appropriately

The software architecture must be developed in conjunction with the systems architecture and software issues must be included in systems engineering tradeoffs.

The architecture and design drivers described in this section include precedence, quality attribute constraints, technology readiness, legacy system considerations, and COTS/government off-the-shelf (GOTS)/reuse components. These drivers can be affected by other drivers. For example, policies and mandates described in Section 3.2.1 can constrain the architecture.

3.5.2.1 Precedence

The degree to which the system is precedented or unprecedented has a significant impact on the risk of a program. According to the *Encyclopedia of Software Engineering, precedented* systems are those for which

- the requirements are consistent and well-understood
- the system architecture (both hardware and software) is known to be adequate for the requirements
- the acquisition and development teams have worked together to build a previous similar system [Marciniak 05]

Violation of one or more of these causes the system to be classified as unprecedented.

Clearly, we are better at producing some types of software than others. We can consistently produce small or common software applications. In fact, a commercial marketplace has developed to provide these software components. On the other hand, we are less capable of producing large, unprecedented systems where few ready-made components are available or where the complexity of creating a unified system from components is stretched past previous limits.

Major stumbling blocks for unprecedented software-critical systems include

- ill-defined, incomplete, contradictory, and changing requirements
- inadequate architecture
- software and hardware architecture mismatches
- lack of domain experience

Relatively simple systems present relatively little risk. Similar systems can be used as benchmarks for reasoning about the system underdevelopment. Unprecedented systems obviously present much greater risk in that there are no equivalent benchmarks from which to draw inferences. A program can only extrapolate from other related systems. The more unprecedented a system, the more risk for the program. The degree to which a system is precedented can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.2.2 Quality Attribute Constraints

The definition of quality attributes was discussed in Section 3.5.1.3. The system can have simplistic qualities and few consequences for failure, very demanding qualities and severe consequences for failure, or be somewhere in between. The quality attributes can put significant constraints on how the system can be architected and implemented. The extent to which a program's quality attribute requirements constrain the solution set can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.2.3 Technology Readiness

Developing software for a system depends on many technologies. Is the hardware mature enough for the software to be developed? Is the development environment including compilers, configuration management tools, and other development tools sufficient to support the software development effort? Developing software also depends on the specific operational context. For example, Java is fairly mature at this point, but there are still some applications where it is inappropriate to use.

The extent to which system technologies are mature enough for software to be developed can be rated on a sliding scale that extends from one extreme to the other: Immature to Mature.

3.5.2.4 Legacy Considerations

Legacy system considerations introduce many avenues for risk during an acquisition including how well the legacy system and its interfaces are documented and managed, the amount of change between a legacy system and its replacement system, backward compatibility requirements, and so on. These are just a few of the many considerations, but they highlight the risks that programs that have legacy requirements must address.

How well the legacy system is understood significantly influences risk. If a legacy system and its interfaces are poorly documented there could be many interoperability and replacement issues. Is the legacy system written in some outdated or obscure programming language? How accurate is the information provided? How hard will it be and how long will it take to obtain any additional information your program requires? Does the government have full data rights to the existing system? How long will you have to interoperate with a legacy system? How will you keep up with the changes to the legacy system?

There are also transition risks as you move from a working legacy system to a new system. Do the two systems need to interoperate for some time or does the new system completely replace the old system? Will there be a new user interface to learn? Are there any safety or security risks as a result? Are there any facilities issues caused by the transition needs?

The requirement to be backward compatible with a prior or legacy system adds its own set of risks, especially when the earlier system still exists and is undergoing changes. How will new requirements in the legacy system be coordinated and managed with the new one? Is an appropriate governance board in place? How different can the new system be from the prior one? How long will backward compatibility need to be maintained after deployment?

The complexity of legacy system aspects that must be considered can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.2.5 COTS, GOTS, and Software Reuse

Maximizing COTS/GOTS/reuse is often a goal in programs today. The implications of using COTS/GOTS/reuse must be fully considered. An acquisition program must have their “eyes wide open” to understand the benefits and mitigate the risks.

COTS or GOTS product use within the system may be considered a low-risk alternative for implementing specific software driven portions of the product design. However, COTS/GOTS should be considered as risky as any other software. The off-the-shelf options should not be considered “low risk” simply because the product can be purchased or obtained as a pre-packaged unit. COTS/GOTS products are not always complete or completely tested and most often are not based on the requirements for the new system being built. Planning for COTS/GOTS use, including a maintenance strategy, is an important part of up-front program planning.

The use of COTS/GOTS can be beneficial but it does bring some challenges with it. Some key challenges include [Carney 03]

- COTS products are driven by the marketplace, not the system context
- built-in assumptions of end-user processes that may not match yours
- licensing, data rights, and warranties
- limited control of content or frequency of releases
- limited visibility into product internals and behavior
- varying architectural paradigms between the product and other system components
- GOTS sustainment responsibilities and plans

The way reuse is defined, planned for (e.g., productivity savings, maintenance), and managed determines the risk involved with reuse. Effective reuse typically involves more than just source code; design, requirements, test scripts can all be reused as well. The degree to which the characteristics of the program’s intended operational context are similar to the operational context for which the original code was intended is one determinant of the risk involved with reuse.

Code reuse can come in many forms including reuse of legacy code and reuse of code between components/modules of the system being acquired. One must also take into account that such reuse is rarely ever “pure” (meaning that the code is reused as-is without any modification). More often the reused code is modified in some way to meet the specific requirements of the system under development or the reused must be “wrapped” by newly developed code to provide a consistent interface to the rest of the system. In either case, there is new development involved with the reuse and the amount of that new development will offset (at least partially) the benefits of the reuse.

The extent to which COTS/GOTS/reuse is planned to provide system functionality and realized over time can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.3 Verification and Test Drivers

The verification and test drivers described in this section include the complexity and availability of the test environment and the number of system configurations.

3.5.3.1 Test Environment Complexity

Typically, as the test environment complexity increases, the risk also increases. Do you need a special environment to perform software testing such as an aircraft? Is a classified test environment required? Is destructive testing required? Is the software part of a self-destruct capability of an end item? How complex are the test tools? Has appropriate training been provided? Is the test environment spread over several geographical locations that must be connected via a network? Are some of the elements of the test environment owned by other entities, for example, a government laboratory? Will simulations (of nascent hardware or of battlefield conditions) need to be developed?

The complexity of the test environment can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.3.2 Test Environment Availability

Typically, as the availability of the necessary test environment decreases, the risk increases. Can you get the testing time required in the necessary test environment? What are the priorities and schedule constraints on the test agency and facilities? How long does it take to get into the test cycle? Does this match up with your schedule? Is another program ahead of you using the test environment? What is the likelihood of slippage into your schedule time?

In addition to the agency and facilities for testing, you also have to consider the components with which you need to test. For example, if you performed some testing that interfaces with an external element that is a prototype or engineering model, what is the risk that the final version of the element will differ from what you tested with?

The availability of the necessary test environment can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.3.3 Number of System Configurations

The number of system configurations has an impact on the test requirements and how they will be tested. Tradeoffs will need to be made regarding the test requirements of the different configurations given the amount of funding available for testing. Can the same test cases be used or do different test cases need to be developed? What about test simulators? Can they be parameterized or do they need to be different?

Even if you only have one configuration under test, ensuring that the configuration is not modified during the testing process is not always easy. For example, a testbed was used during the day for testing the system configuration on a project. In the evening it was used for training purposes. Between the daily testing cycles, the test system configuration was changed by the training team without the test team's knowledge. How valid were the tests? The importance of managing the configuration of the system, test environment, and other test artifacts is essential.

The number of system configurations that need to be tested can be rated on a sliding scale that extends from one extreme to the other: Few to Many.

3.5.4 Deployment Drivers

The deployment drivers described in this section include the number of sites and system configurations, user and maintainer readiness, and data migration considerations.

3.5.4.1 Number of Sites

Sites may have different environments (hardware, software, networking, etc.) that impact the deployment. Personnel may have different levels of expertise at different sites. The approval process or acceptance process may be different. Another factor to consider is the timing for the deployments if there are multiple sites. If there is schedule overlap, staffing considerations must be taken into account.

The number of sites the system will be deployed to can be rated on a sliding scale that extends from one extreme to the other: Few to Many.

3.5.4.2 User Readiness

How ready the users are to accept and use the system is critical to the success of the system in fulfilling the mission. Some software considerations include

- Have all operational software training needs been identified?
- Will any new training facilities or simulators be needed?
- Does the life cycle support the development of software for training assets?
- Are the organizations impacted involved throughout the life cycle?
- Will there be changes in personnel allocations to operate/maintain the system?

The extent to which users have the skills and desire to use the system can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.4.3 Maintainer Readiness

How ready the maintainers are to operate and maintain the system is also critical to the mission. Some software considerations include

- Have all maintenance software training needs identified?
- Will any new tools be needed? Does the life cycle support the development of software for the new tools?
- Does the life cycle support the development of software for training assets?
- Are the organizations impacted involved throughout the life cycle?

The extent to which maintainers have the skills, tools, and desire to operate and maintain the system can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.4.4 Transition and Data Migration

Data migration, data base schema and data dictionary changes, access modifications, and other technical transition issues can increase risk in an acquisition program. As the number of new or changing technical elements increases, risk can compound. How will the transition be staged? Will parallel operations between the old applications or databases and the new ones be needed? Will data be migrated manually or will it be automated? How will the integrity of the data and the system operations be verified? Is the support infrastructure in place to support the change? Is there a contingency plan?

The extent that technical transition issues need to be addressed can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.5 Maintenance and Support Drivers

The maintenance and support drivers described in detail in this section include the number of system configurations, update readiness, support duration, re-competition readiness, operational environment, legacy system support considerations and availability of data rights.

3.5.5.1 Number of System Configurations

Does the system require different software for different sites? If so, how many configurations need to be supported? The number of system configuration that will be deployed can be rated on a sliding scale that extends from one extreme to the other: Few to Many.

3.5.5.2 Update Readiness

System updates are inevitable. How frequently the system is updated, when, how, and by who are all factors that must be considered. Usually, as the number of updates increases, so does the risk. But there are other risk factors to consider when updating software. For example, lack of an adequate configuration control process for software changes could cause havoc even if only one software update was applied. Other factors include

- Is software maintenance adequately planned and emphasized?
- Is the software maintainable?
- Can the system and software architecture support future changes?
- Are support assets such as training and documentation being updated concurrently?
- Are the software development environment and test tools available to the maintainers?

The extent to which the program is ready to update the system can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.5.3 Support Duration

The length of time the system will be in sustainment must be considered when developing and modifying your acquisition strategy. What effort is required to avoid decay and obsolescence? How will obsolescence of COTS products or other enabling technologies be planned and managed? How will you ensure support continuity? Is it better to use your development contractor or a depot? If the system was initially developed with a 5-year window and now you are asked to sustain it for 10 years are you going to change anything? Do you have a system maintenance plan?

The length of time the system is planned to be in sustainment can be rated on a sliding scale that extends from one extreme to the other: Short to Long.

3.5.5.4 Re-Competition Readiness

If your acquisition strategy includes successive competitions or you are forced into a competitive situation (i.e., due to performance) will you be ready to re-compete the contract when the time comes? Is system documentation up-to-date? Are the technologies supportable? For example, when the Y2K crisis arose, there was great demand for support options that were in limited supply. How will support options and their availability change during contingency operations?

The extent to which you are ready to re-compete your acquisition in whole or part can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.5.5 Operational Environment

The environment in which the system will operate can have an impact on your acquisition strategy. The operational environment can include the natural environment (e.g., space or desert), user conditions (e.g., immigrant workers with well worn fingerprints crossing through a border checkpoint that uses fingerprints as an identification mechanism), and requirements of the computer or other devices the software interacts with.

You must understand the conditions or constraints under which the system will operate. For example, does the software test for boundary conditions related to the environment?

The constraints of the operational environment can be rated on a sliding scale that extends from one extreme to the other: Benign to Harsh.

3.5.5.6 Legacy Considerations

Legacy system considerations were discussed in Section 3.5.2.4. During the maintenance and support part of the life cycle, additional legacy system considerations must be taken into account. What happens if a legacy system your system depends on will no longer be supported? Is the maintenance schedule dependent on a legacy system update or vice versa? Are there interface changes that need to be supported?

The complexity of the legacy system aspects that must be considered during maintenance and support can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.5.7 Complexity of Data Rights

The availability of data rights is a complex aspect of acquisition, particularly as systems continue to grow in size and their use of COTS software, GOTS software, free and open source software, and more. The Council on Governmental Relations[2] defines *data rights* as the "license the government obtains in technical data which grantees and contractors deliver to the government after completion of the project." It goes on to state that what makes it so complex are the gradations in scope from non-exclusive to exclusive license rights. There are also differences between how DoD and civilian agencies determine allocation of rights and the definition of technical data rights versus the term "computer software." In addition, FAR Subpart 27.4 is devoted to data rights.

The complexity of data rights issues that must be considered can be rated on a sliding scale that extends from one extreme to the other: Low to High.

3.5.6 Disposal Driver

It might not be readily apparent that software has an impact on disposal, but there are some disposal issues with software. Classification is probably the largest concern. Are there any classification concerns with the software in the system? Another issue is the ability to remove software from the system. How will it be removed? Will its removal affect other systems operating on the same hardware?

The actions taken during the acquisition can make the activities to ensure the disposal of decommissioned, destroyed, or irreparable system components meet all applicable regulations

[2] For more information, visit http://www.cogr.edu/docs/RightsComputer.htm.

less or more difficult. As the acquisition strategy evolves over time, software implications for disposal should not be ignored.

The extent to which a program has software disposal considerations can be rated on a sliding scale that extends from: Unrestricted to Restricted.

4 Key Acquisition Strategy Elements

4.1 Strategy Element Overview

All strategies can be plans, but not all plans can be strategies. In the context of acquisition, *strategies* are high-level decisions that direct the development of more detailed plans that guide the execution of a program. Careful planning of what is to be done, who will do it, and when it will be done is required.

Developing an all-encompassing acquisition strategy for a program is a daunting activity. As with many complex endeavors, often the best way to begin is to break the complex activity down into simpler, more manageable tasks. Our approach to developing an acquisition strategy is not to craft a monolithic plan, but to a carefully integrate a collection of individual strategy elements and corresponding strategic choices tailored to address the program drivers. Thus, when developing an acquisition strategy, a program manager's first task is to define the elements of that strategy. When defining strategy elements, it is useful to ask the question, "What acquisition choices must I make in structuring this program?" Inevitably, asking this question leads to more detailed questions such as

- What acquisition approach should I use?
- What type of solicitation will work best?
- How will I monitor my contractor's activities?
- and many more

The result of these choices defines the acquisition strategy. Identifying the strategy elements is the first step in this process. Several resources exist to aid us in this task.

The Defense Acquisition University's (DAU) *Defense Acquisition Guidebook* defines 19 "strategy considerations" that should be addressed when formulating an acquisition strategy [DAU 04]. Many of these considerations, listed in Table 3, are composed of lower-level supporting considerations. Each consideration requires careful attention by the program management team and should be addressed within the acquisition strategy.

Table 3: DAU Defense Acquisition Guidebook Acquisition Strategy Considerations

Acquisition Strategy Consideration	Supporting Considerations	
Program Structure	• Milestone Decision Points	• Acquisition Phases
Acquisition Approach		
Capability Needs		
Test and Evaluation		
Risk Management	• Risk Management in Systems Engineering	• Risk Management in Program Protection
Resource Management	• Cost Estimation • Funding	• Advance Procurement
Systems Engineering Plan	• Technical Baseline Management • Technical Reviews	• Metrics
Interoperability	• Information Interoperability	• Other Interoperability (intersystem, interprogram)
Information Technology	• Infrastructure Considerations	• Support Considerations
Research and Technology Protection	• Protection of Critical Program Information	• Anti-Tamper Measures
Information Assurance		
Product Support Strategy	• Product Support (including software) • Interoperability • Data Management (DM) • Integrated Supply Chain Management • Life Cycle Cost Optimization • Logistics Footprint Minimization • Life Cycle Assessment • Demilitarization and Disposal	• Environment, Safety, and Occupational Health (ESOH) • Human Systems Integration (HSI) • Maintain Designated Science and Technology Information, the Security Classification Guide, and the Counter-intelligence Support Plan.
Human Systems Integration	• Manpower • Personnel • Training • Human Factors	• Safety and Occupational Health • Personnel Survivability • Habitability
Environment, Safety, and Occupational Health (ESOH)		
Modular Open Systems Approach (MOSA)	• MOSA Integration • MOSA Development	• MOSA Monitoring
Business Considerations	• Competition • International Cooperation • Contract Approach	• Leasing • Equipment Valuation
Best Practices	• Integrated Product and Process Development • Performance-Based Specifications • Management Goals • Reporting and Incentives • Modular Open Systems Approach • Replacement of Government-Unique Management and Manufacturing Systems • Technology Insertion for Continuous Affordability Improvement	• Realistic Cost Estimates and Cost Objectives • Adequate Competition Among Viable Offerors • Best Value Evaluation and Award Criteria • Use of Past Performance in Source Selection • Software Capability E\valuations • Government-Industry Partnerships
Relief, Exemption, or Waiver		
Additional Acquisition Strategy Topics	• Program Office Staffing and Support Contractor Resources Available to the Program Manager • Integrated Digital Environment Management • Government Property in the Possession of Contractors Management	• Simulation Based Acquisition and Modeling and Simulation • Software-Intensive Programs Review

Another resource is the *CMMI Acquisition Module (CMMI-AM)*, *Version 1.1* [Bernard 05]. As shown in Table 4, this interpretive guide to the use of the CMMI framework in an acquisition environment defines 12 Process Areas to be addressed by the acquirer. Each of these process areas should also be addressed by the acquisition strategy.

Table 4: CMMI-AM and Acquisition Strategy Elements

CMMI-AM Process Area	Goals
Project Planning	• Estimates of project planning parameters are established and maintained. • A project plan is established and maintained as the basis for managing the project. • Commitments to the project plan are established and maintained.
Project Monitoring and Control	• Actual performance and progress of the project are monitored against the project plan. • Corrective actions are managed to closure when the project's performance or results deviate significantly from the plan. • Work is coordinated with suppliers to ensure the contract is executed properly.
Solicitation and Contract Monitoring	• The project is prepared to conduct the solicitation. • Suppliers are selected based on the solicitation package. • Contracts are issued based on the needs of the acquisition and the suppliers' proposed approaches.
Integrated Project Management	• The project is conducted using a defined process that is tailored from the organization's set of standard processes. • Coordination and collaboration of the project with relevant stakeholders are conducted.
Risk Management	• Preparation for risk management is conducted. • Risks are identified and analyzed to determine their relative importance. • Risks are handled and mitigated, where appropriate, to reduce adverse impacts on achieving objectives.
Requirements Development	• Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements. • Customer requirements are refined and elaborated to develop product and product/component requirements. • The requirements are analyzed and validated, and a definition of required functionality is developed.
Requirements Management	• Requirements are managed and inconsistencies with project plans and work products are identified.
Verification	• Preparation for verification is conducted. • Peer reviews are performed on selected work products. • Selected work products are verified against their specified requirements.
Validation	• Preparation for validation is conducted. • The product or product components are validated to ensure that they are suitable for use in their intended operating environment.
Decision Analysis and Resolution	• Decisions are based on an evaluation of alternatives using established criteria.
Measurement and Analysis	• Measurement objectives and activities are aligned with identified information needs and objectives. • Measurement results that address identified information needs and objectives are provided.
Transition to Operations and Support	• Preparation for transition to operations and support is conducted. • Transition decisions and actions are executed in accordance with transition criteria.

By examining the strategy considerations shown in Table 3, we find a mixture of both strategy drivers and elements of an acquisition strategy. For example, the Acquisition Approach is a decision that must be made by the program manager—it is a strategy element. Similarly, the Product Support Strategy is a collection of decisions to be made—it is a strategy element. Other considerations, such as Interoperability and Capability Needs are not necessarily decisions to be made by the program manager, but are factors that influence the execution of the project; therefore, they are program drivers.

Likewise, the CMMI-AM outlined in Table 4 does not clearly identify acquisition strategy elements. It lists the goals to be accomplished, but says nothing about the strategies employed to accomplish them.

While both the *Defense Acquisition Guidebook* and the *CMMI-AM* are helpful in framing the concepts behind developing an acquisition strategy, neither explicitly defines nor differentiates the drivers and the elements of an acquisition strategy in a manner suitable for using a specific *method* to develop a strategy. However, using both of these resources as a reference, we have created a partial list of strategy elements that could be applied methodically to strategy development. These elements are shown in Table 5; boldfaced text denotes the elements and sub-elements discussed in this report.

Table 5: Partial List of Strategy Elements and Sub-Elements

Strategy Element	Strategy Sub-Element
Program Structure	Milestone Decision Points
	Acquisition Phases
Acquisition Approach	
Business Considerations	**Competition**
	Solicitation
	Source Selection
	Contract Approach
Risk Management	
Information Assurance	**Supplier Assurance**
Test and Evaluation	
Product Support	**Training**
	Installation
	Source of Support

This remainder of this section describes three strategy elements and their sub-elements (Acquisition Approach, Business Considerations, and Product Support) in more detail. We will later use these strategy elements illustrate the strategy development method. For each of these three strategy elements and sub-elements, we define the element (or sub-element), discuss the range of strategy choices, and provide examples of those choices. Then, using the categories of software characteristics and subsequent strategy drivers documented in Section 3 (and summarized in Table 6), we present a “strategy profile” for each strategy element or subelement. The *strategy profile* summarizes the principle drivers that influence a strategy element and are presented in Table 7–Table 13.

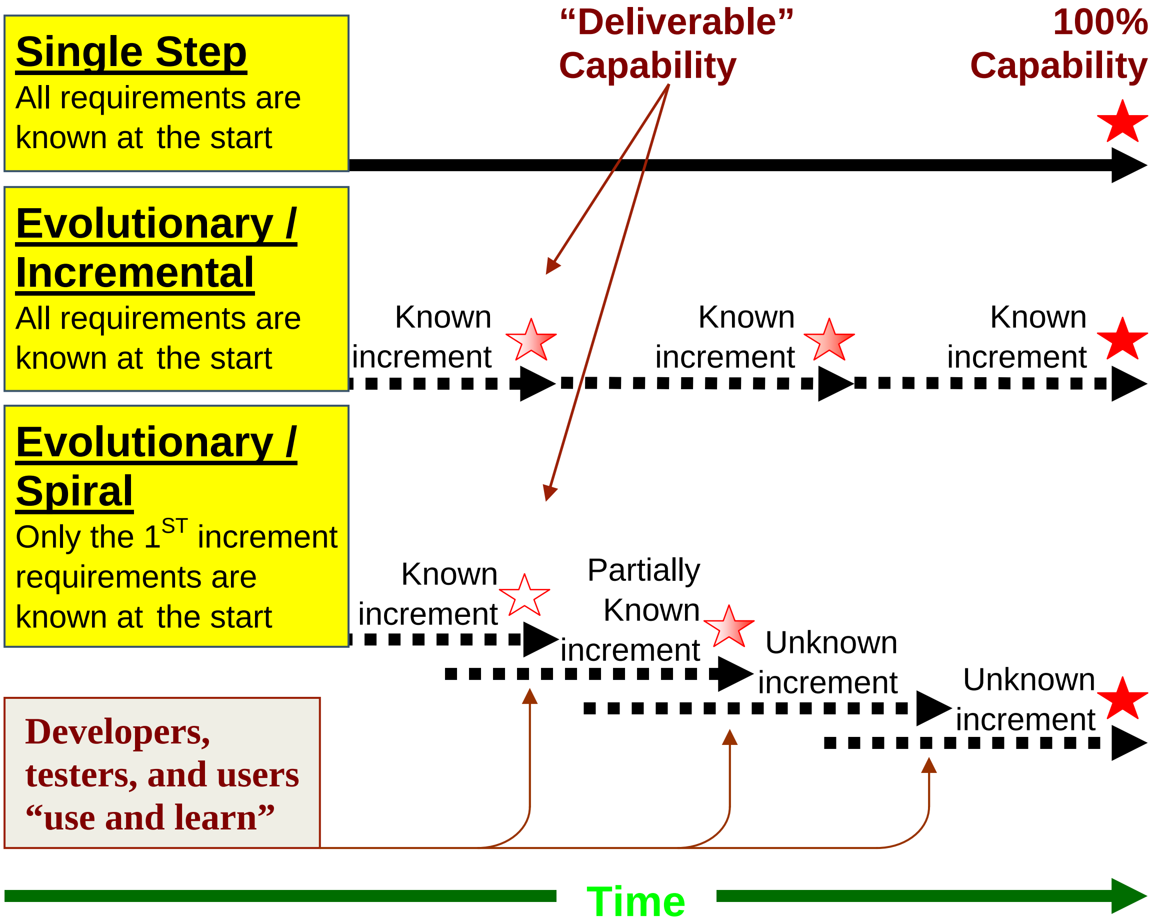
Table 6: Strategy Driver Taxonomy

1. **Software Criticality**
2. **Acquisition Environment Category Drivers**
   - a. Policies and Mandates
   - b. Supplier Availability
3. **Programmatic Category Drivers**
   - a. Mission Needs and Scope
   - b. Funding
     - *i. Funding Constraints*
     - *ii. Funding Profile*
   - c. Schedule
     - *i. Schedule Constraints*
     - *ii. Urgency*
4. **Organizational Category Drivers**
   - a. Program Management Office Capabilities
     - *i. PMO Staff Skills*
     - *ii. PMO Staff Capacity*
     - *iii. PMO Staff Stability*
     - *iv. PMO Process Focus*
   - b. Stakeholders
     - *i. Number and Diversity*
     - *ii. Level of Engagement (responsiveness and quality)*
     - *iii. Level of Agreement*
   - c. Supplier Capability
     - *i. Supplier Staff Skills*
     - *ii. Supplier Staff Capacity*
     - *iii. Supplier Staff Stability*
     - *iv. Supplier Performance to Date*
5. **Life Cycle Category Drivers**
   - a. Product Definition and Specification
     - *i. Requirements Volatility*
     - *ii. Requirements Understanding*
     - *iii. Quality Attribute Definition Quality*
     - *iv. Interoperability*
   - b. Architecture and Design
     - *i. Precedence*
     - *ii. Quality Attribute Constraints*
     - *iii. Technology Readiness*
     - *iv. Legacy Considerations*
     - *v. COTS / GOTS / Reuse*
   - c. Verification and Test
     - *i. Test Environment Complexity*
     - *ii. Test Environment Availability*
     - *iii. Number of System Configurations*
   - d. Deployment
     - *i. Number of Sites*
     - *ii. User Readiness*
     - *iii. Maintainer Readiness*
     - *iv. Transition / Data Migration*
   - e. Maintenance and Support
     - *i. Number of System Configurations*
     - *ii. Update Readiness*
     - *iii. Support Duration*
     - *iv. Re-Competition Readiness*
     - *v. Operational Environment*
     - *vi. Legacy Considerations*
     - *vii. Complexity of Data Rights*
   - f. Disposal

4.2 Acquisition Approach Strategy Element

Note: Some of the information contained in this section is familiar to most experienced acquisition PMO staff. It is reiterated in this report to lend coherence to the authors' discussion of acquisition strategy elements.

The Acquisition Approach strategy element defines the approach the program will use to achieve full capability. Typically, this approach is either single step, evolutionary/incremental, or evolutionary/spiral, as illustrated in Figure 9.


Single Step
All requirements are known at the start
"Deliverable" Capability
100% Capability
Evolutionary / Incremental
All requirements are known at the start
Known increment
Known increment
Known increment
Evolutionary / Spiral
Only the 1ST increment requirements are known at the start
Known increment
Partially Known increment
Unknown increment
Unknown increment
Developers, testers, and users "use and learn"
Time

Based on AF Program Manager Workshop presented by Mr. Little

Figure 9: Defining an Acquisition Approach

4.2.1 Single-Step Approach

As the name implies, the single-step acquisition approach consists of the acquisition of a defined capability in one increment. The delivered items provide 100% of the required capability. To achieve this result, the acquirer must know all of the requirements at the start of the program. Of course, virtually no program proceeds to completion without the addition or modification of some requirements; however, many programs do start with a firm definition of the goal and a clear plan on how to attain it. For these types of programs, requirements changes are typically less significant.

The single-step acquisition approach is most appropriate for programs that have the following characteristics:

- The capability is clearly defined.
- All of the requirements, including the software requirements, are known.
- The technology to achieve the capability is mature.
- There is no demand for earlier deployment of partial capability.
- There are successful precedents for the program (i.e., other similar programs delivering similar capabilities have been successfully completed).

4.2.2 Evolutionary/Incremental Approach

The evolutionary/incremental acquisition approach consists of the acquisition of a defined capability in multiple, defined increments. Each increment provides a defined and "fieldable" capability. Initial increments may provide only a portion of the total capability required with each increment adding more capability until full capability is achieved. Because the capabilities delivered in each increment are pre-defined during the acquisition planning process, again, the acquirer must know all of the requirements at the start of the program.

The evolutionary/incremental acquisition approach provides more flexibility than the single-step method; it enables the acquirer to address issues arising from either technology maturity or user needs for early fielding of some capabilities. This flexibility may come at the expense of more effort for the acquirer, more cost, and/or longer schedules.

The evolutionary/incremental acquisition approach is most appropriate for programs that have the following characteristics:

- The capability is clearly defined.
- All of the requirements, including the software requirements, are known.
- The technology to achieve the capability in the first increment is mature. The technologies to achieve the capabilities of the later increments are proven, but require further development before deployment.
- There is a demand for earlier deployment of some partial capabilities.
- There are successful precedents for the program (i.e., other similar programs delivering similar capabilities have been successfully completed).

4.2.3 Evolutionary/Spiral Approach

The focus of the evolutionary/spiral acquisition approach is risk mitigation and risk reduction. This approach consists of the acquisition of a defined capability in multiple, undefined increments. Each increment is often referred to as a "spiral." At the beginning of each increment, the acquirer defines a “deliverable” capability for that increment. Initial increments focus on the aspects of the capability that inject the greatest risk into the program. These are usually aspects associated with unprecedented capabilities and new, immature technologies. At the end of each increment, the results of the increment are reviewed and a risk analysis of the overall program is performed. Based on this evaluation, the next increment is defined. Only after these risky endeavors have been completed does the acquirer begin to address the more conventional, less risky aspects of the program. In this manner, a “showstopping” technological issue can be identified early in the program. This enables the acquirer to consider alternative approaches before significant “sunk costs” are incurred.

To successfully execute an evolutionary/spiral acquisition, the acquirer first needs a clear understanding of the final capability that is desired. This serves as the target for the successful acquisition spirals. Driven by a risk analysis of the program, the acquirer identifies its most significant risks and forms a risk reduction strategy to address them. Such a strategy can result in early increments that involve

- evaluating candidate system components in the context of the program
- creating laboratory or pre-production prototypes of challenging components
- transitioning immature technologies from laboratory to the program environment

Subsequent increments build on the earlier increments.

Because the capabilities delivered in each increment are defined at the initiation of that increment, the acquirer does not need to know all of the system capabilities at the initiation of the program. He or she must only know all of the requirements for the next increment at the start of that increment. Requirements for the next increment are defined only during and after execution of the current increment.

In this manner, the evolutionary/spiral acquisition approach provides considerably more flexibility than the single-step method, enabling the acquirer to address high-risk issues within the program. This flexibility may come at the expense of more effort for the acquirer, more cost, and/or longer schedules.

Key points that are crucial to the success of spiral acquisition include

- The acquirer needs to obtain a clear understanding of the final capability that is needed.
- Increments should be risk-driven; each increment should be defined to reduce the highest risks facing the program at that time.
- At the start of each spiral, the acquisition plan should be revisited to fully specify the capabilities to be delivered in that spiral and to tentatively define the future spirals leading to final capability.
- Spirals should have well defined exit points.

The evolutionary/spiral acquisition approach is most appropriate for programs that have the following characteristics:

- The capability is not fully defined or the stakeholders have not reached consensus on the capability definition.
- All of the requirements, including the software requirements, are not known (but requirements for the first increment are known).
- All of the technology needed to achieve the final capability is not fully mature (but the technology required for the first increment is mature).
- There may be a demand for earlier deployment of some partial capabilities.
- There are not successful precedents for the program (i.e., other similar programs have not delivered similar capabilities).

Unlike many strategy elements, the Acquisition Approach strategy element has a defined range of choices consisting of only the three options presented above. As discussed, the key program drivers influencing the choice of an acquisition approach is primarily influenced by the completeness of the requirements, requirements volatility, and the demand for interim capability delivery.

The strategic choices and drivers of the Acquisition Approach strategy element are summarized in the strategy profile shown in Table 7.

Table 7: Acquisition Approach Strategy Profile

Strategy Element Range	Single-Step Evolutionary / Incremental Evolutionary / Spiral
Principal driver influences (from Table 6)	**1. Software Criticality** **2. Acquisition Environment Cat. Drivers** a. Policies and Mandates b. Supplier Availability **3. Programmatic Category Drivers** a. Mission Needs and Scope b. Funding *i. Funding Constraints* *ii. Funding Profile* c. Schedule *i. Schedule Constraints* *ii. Urgency* **4. Organizational Category Drivers** a. Program Management Office Capabilities *i. PMO Staff Skills* *ii. PMO Staff Capacity* *iii. PMO Staff Stability* b. Stakeholders *i. Number and Diversity* *ii. Level of Engagement (responsiveness and quality)* *iii. Level of Agreement* **5. Life Cycle Category Drivers** a. Product Definition and Specification *i. Requirements Volatility* *ii. Requirements Understanding* *iii. Quality Attribute Definition Quality* *iv. Interoperability* b. Architecture and Design *i. Precedence* *ii. Quality Attribute Constraints* *iii. Technology Readiness* *iv. Legacy Considerations* *v. COTS / GOTS / Reuse*

4.3 Business Considerations: Competition Strategy Element

Note: Some of the information contained in this section is familiar to most experienced acquisition PMO staff. It is reiterated in this report to lend coherence to the authors' discussion of acquisition strategy elements.

Competition is a key element of the acquisition strategy; it is recognized as a means of achieving the best acquisition value. The acquirer should also use it as a means of fostering innovation for defense applications. During competition, potential suppliers analyze the needs of the acquirer and provide their best ideas to address these needs. More competition results in more ideas, some of which may be innovative and valuable to the acquirer.

Competition options (strategic choices) for the acquirer include

- Full and Open Competition
- Full and Open Competition After Exclusion of Sources
- Sole Source Contracting

The default competition strategy is full and open competition; however, Federal Acquisition Regulations (FAR) Part 6, Subpart 6.3[3] defines the circumstances under which full and open competition may be waived. Full and open competition may be implemented through sealed bids, competitive proposals, combinations of competitive procedures (e.g., two-step sealed bidding), or other competitive procedures as described in FAR Part 6, Subpart 6.1.

Full and open competition after exclusion of sources enables the acquirer to restrict the suppliers who may respond to the solicitation. After the exclusion of sources, full and open competition is implemented for the remaining sources. An acquirer may exclude potential sources to

- establish or maintain alternative sources (i.e., exclude bidders to provide opportunity for other bidders to enter the market) (FAR 6.202)
- limit the bidders to Small Business Concerns (FAR 6.203)
- execute the acquisition through the Small Business Administration, again limiting the bidders to Small Business Concerns (FAR 6.204)
- limit the bidders to HUBZone Small Business Concerns (FAR 6.205)
- limit the bidders to Service-Disabled Veteran-Owned Small Business Concerns (FAR 6.206)

Sole source competition enables the acquirer to award a contract without full and open competition. This is permitted only under the following extraordinary conditions:

- Only One Responsible Source and No Other Supplies or Services Will Satisfy Agency Requirements (FAR 6.302-1)
- Unusual and Compelling Urgency (FAR 6.302-2)
- Industrial Mobilization; Engineering, Developmental, or Research Capability; or Expert Services (FAR 6.302-3)
- International Agreement (FAR 6.302-4)
- Authorized or Required by Statute (FAR 6.302-5)
- National Security (FAR 6.302-6)
- Public Interest (FAR 6.302-7)

[3] Throughout this report, a number of references are made to the Federal Acquisition Regulations (FARs). Additional regulations such as the Defense Federal Acquisition Regulations Supplement (DFARS), Army Federal Acquisition Regulations Supplement (AFARS), Air Force Federal Acquisition Regulations Supplement (AFFARS), may also apply.

The acquirer must define the competition planned for all phases of the program’s life cycle (e.g., Concept Refinement, Technology Development, System Development and Demonstration, Production and Deployment, and Operations and Support). Different competition strategies may be used in these life-cycle phases. For example, full and open competition may be used in Concept Refinement and Technology Development phases, with sole source contracting used for later phases. When utilizing evolutionary (either incremental or spiral) acquisition, the acquirer should evaluate the costs and the benefits of competing each increment.

Competition must also be considered when contracting for support of the product. If support is not contracted as part of system development, the acquirer must ensure that appropriate technical data packages and appropriate data rights are procured during the development phase to support competition and contracting of support.

Factors that influence the Competition strategy elements include

- available sources of supply
- small business set-aside statutes
- special circumstances (e.g., only one responsible source, urgency, industrial mobilization, international agreement, other statutory requirement, national security, public interest)

The strategic choices and drivers of the Competition strategy element are summarized in the strategy profile shown in Table 8.

Table 8: Competition Strategy Profile

Strategy Element Range	Full and Open Competition Full and Open Competition After Exclusion of Sources Sole Source Contracting
Principal driver influences (from Table 6)	**1. Software Criticality** **2. Acquisition Environment Cat. Drivers** a. Policies and Mandates b. Supplier Availability **3. Programmatic Category Drivers** a. Mission Needs and Scope b. Funding *i. Funding Constraints* *ii. Funding Profile* c. Schedule *i. Schedule Constraints* *ii. Urgency* **4. Organizational Category Drivers** a. Program Management Office Capabilities *i. PMO Staff Skills* *ii. PMO Staff Capacity* *iii. PMO Staff Stability* **5. Life Cycle Category Drivers** a. Product Definition and Specification *i. Requirements Volatility* *ii. Requirements Understanding* *iii. Quality Attribute Definition Quality* *iv. Interoperability* b. Architecture and Design *i. Precedence* *ii. Quality Attribute Constraints* *iii. Technology Readiness* *iv. Legacy Considerations* *v. COTS / GOTS / Reuse* c. Verification and Test *i. Test Environment Complexity* *ii. Test Environment Availability* *iii. Number of System Configurations* d. Deployment *i. Number of Sites* *ii. User Readiness* *iii. Maintainer Readiness* *iv. Transition / Data Migration*

4.4 Business Considerations: Solicitation Strategy Element

Note: Some of the information contained in this section is familiar to most experienced acquisition PMO staff. It is reiterated in this report to lend coherence to the authors' discussion of acquisition strategy elements.

A *solicitation* is often the acquirer's first formal attempt to transmit his needs to the constellation of potential suppliers. It is also often the first time that many suppliers become aware of the proposed acquisition. Typical forms of solicitation include the following:

- **Invitation for Bid (IFB)** – An IFB is a solicitation issued by the acquirer to the community of potential bidders. An IFB describes the needs of the acquirer (i.e., the requirements) and the criteria for the evaluation of the offerors' bids and proposals. Contract award is made without negotiation and is based on the lowest bid. This solicitation method is not typically suitable for the acquisition of complex, custom-built, software intensive systems.
- **Request for Proposal (RFP)** – An RFP is a solicitation issued by the acquirer to the community of potential bidders. An RFP describes the needs of the acquirer (i.e., the requirements) and the criteria for the evaluation of the offerors' bids and proposals. Evaluation criteria typically include factors such as price, schedule, technical merit, management capability, risk, and so on. Contract award is based upon evaluation of specified factors. Negotiations can be conducted with offerors.

  An RFP can be issued in combination with a Statement of Work (SOW) or Statement of Objectives (SOO). A SOW provides a detailed definition of the scope of the contract; whereas, a SOO provides a more general statement of the objectives of the contract. For example, a SOW may be used to define the requirements for an air-to-air missile. A SOO may be used to define the objectives of destroying airborne targets of defined characteristics within a defined range. These objectives could be met with a missile, artillery, directed energy weapons, etc.

  For complex systems and/or large programs, it is common to issue an RFP several times in a draft form as a means of soliciting input and feedback from the community of potential bidders. Not only does this process improve the quality of the RFP, it also helps solidify the acquirers' concepts of what is required, and helps the potential bidders to form a more complete understanding of the acquirers needs.
- **Request for Quotation (RFQ)** – An RFQ is a request for information made by the acquirer to the community of prospective suppliers. It is typically used for planning purposes. It is not a commitment to procure the requested product. As a result of the response to an RFQ, the acquirer can refine the product requirements before issuing an IFB or RFP.
- **Request for Information (RFI)** – An RFI is similar to an RFQ in that it is used to solicit knowledge and information from the community of prospective suppliers in the earliest phases of an acquisition. Whereas the RFQ is seeking primarily cost and schedule information to be used for acquisition planning purposes, the RFI is often seeking technical information regarding program feasibility, technology maturity, bidder capabilities, etc.

The strategic choices and drivers of the Solicitation strategy element are summarized in the strategy profile shown in Table 9.

Table 9: Solicitation Strategy Profile

Strategy Element Range	Invitation for Bid (IFB) Request for Proposal (RFP) with SOW Request for Proposal (RFP) with SOO Request for Quotation (RFQ)
Principal driver influences (from Table 6)	**1. Software Criticality** **2. Acquisition Environment Cat. Drivers** a. Policies and Mandates b. Supplier Availability **3. Programmatic Category Drivers** a. Mission Needs and Scope b. Funding *i. Funding Constraints* c. Schedule *i. Schedule Constraints* *ii. Urgency* **4. Organizational Category Drivers** a. Program Management Office Capabilities *i. PMO Staff Skills* *ii. PMO Staff Capacity* *iii. PMO Staff Stability* **5. Life Cycle Category Drivers** a. Product Definition and Specification *i. Requirements Volatility* *ii. Requirements Understanding* *iii. Quality Attribute Definition Quality* *iv. Interoperability* b. Architecture and Design *i. Precedence* *ii. Quality Attribute Constraints* *iii. Technology Readiness* *iv. Legacy Considerations* *v. COTS / GOTS / Reuse*

4.5 Business Considerations: Contract Approach Strategy Element

Note: Some of the information contained in this section is familiar to most experienced acquisition PMO staff. It is reiterated in this report to lend coherence to the authors' discussion of acquisition strategy elements.

A *contract* defines the relationship between the acquirer and the supplier. The acquirer must choose the type of contract to issue for each acquisition. Contract types can be categorized as

1. Fixed-Price Contracts (FAR 16.2)
2. Cost Contracts (FAR 16.302)
3. Incentive Contracts (FAR 16.4)
4. Indefinite-Delivery Contracts (FAR 16.5)
5. Time-and-Materials, Labor-Hour, and Letter Contracts (FAR 16.6)
6. Agreements (FAR 16.7)

The first three of contract categories are most applicable to acquisition of developmental items and are addressed in this section. This section will begin with a definition of the various contract types and then addresses when a particular contract type might be applicable.

4.5.1 Fixed-Price Contracts (FAR 16.2)

Fixed-price contracts provide a defined deliverable in return for a defined price and are usually considered the default contract type. Six variations of fixed-price contracts are discussed below.

- For a Firm-Fixed-Price (FFP) Contract (FAR 16.202), the price is not subject to any adjustment on the basis of the contractor's costs. FFP contracts are most appropriate for acquisition of commercial items. They are also appropriate for acquisition of developmental items with stable and detailed specifications if the acquirer can identify a fair and reasonable price using methods such as price competition, prior purchase history, and so on.
- A Fixed-Price Contract with Economic Price Adjustment (FAR 16.203) also establishes a fixed price; however, it may be adjusted (up or down) in response to specified contingencies such as
  - changes in the market price of specific items used in the execution of the contract
  - changes in specified costs of labor or material experienced by the contractor
  - changes in specified labor or material cost standards or indexes

  This contract type is used when there is significant doubt regarding the stability of the economic conditions throughout the duration of the contract.
- A Fixed-Price Contract with Prospective Price Redetermination (FAR 16.205) provides a firm, fixed price for an initial period of the contract and a plan for redetermination, at defined times during contract execution, of the price for subsequent periods. The contract typically includes a cap on future price determinations. This type of contract is used when the acquirer can negotiate a fair and reasonable FFP for the initial period, but not for later periods.
- A Fixed-Ceiling-Price Contract with Retroactive Price Redetermination (FAR 16.206) establishes a fixed ceiling price and provides for price redetermination within the ceiling after contract completion. This type of contract is used for small research and development contracts if a fair and reasonable firm, fixed price cannot be established.
- A Firm-Fixed-Price, Level-of-Effort Term Contract (FAR 16.207) provides a specified level of effort, over a stated period of time, on work that can only be defined in general terms, in return for a fixed dollar value. This type of contract is typically used for study of a specific topic.

4.5.2 Cost-Reimbursement Contracts (FAR 16.3)

Cost-reimbursement types of contracts pay for costs incurred by the supplier during the execution of the contract. These contracts establish a cost ceiling and obligate funds based upon an estimate of total cost. They are appropriate for contracts where significant uncertainties preclude the possibility of accurately estimating expected costs. Three variations of cost-reimbursement contracts are discussed below.

- A Cost Contract (FAR 16.302) provides reimbursement of the contractors' costs, but the contractor receives no fee. This type of contract is usually applied to research work performed by non-profit organizations.
- A Cost-Sharing Contract (FAR 16.303) provides reimbursement of only a portion of the contractors' costs. Again, the contractor receives no fee. This type of contract is used when the contractor is willing to absorb some of the contract costs in the expectation of some future benefit (e.g., entry into a new market, development of a new technology, improved competitive position for future acquisitions).
- A Cost-Plus-Fixed-Fee Contract (FAR 16.306) provides reimbursement of the contractors' costs. Additionally, the contractor receives a fixed fee determined at contract award. This type of contract is typically used on small contracts for which expected costs cannot be accurately estimated.

4.5.3 Incentive Contracts (FAR 16.4)

Incentive contracts reimburse the costs of the contractor and also provide a fee. Like cost-reimbursement contracts, contract performance targets (e.g. cost targets, schedule targets, technical performance targets) are defined. Unlike the cost reimbursement contracts, the fee is structured in a way to incentivize the contractor to meet these targets. Four variations of incentive contracts are discussed below:

- For a Fixed-Price Incentive Contract (FAR 16.403), the acquirer and contractor negotiate a target cost, a fee schedule, and a ceiling price. The fee varies inversely to the relationship between the actual performance and the performance targets (e.g., greater cost $\Rightarrow$ smaller fee, later delivery $\Rightarrow$ smaller fee, less technical performance $\Rightarrow$ smaller fee). This type of contract is used where the level of cost uncertainty is too great for a fixed-price contract; however, the costs can be bounded sufficiently to establish a ceiling acceptable to both parties. The incentive fee then motivates the contractor to perform efficiently within this boundary.
- A Fixed-Price Contract with Award Fees (FAR 16.404) is a fixed-price contract. Award fees are added to the contract value to motivate contractor performance. Award fees are paid based upon the acquirer's evaluation of the contractor's performance. This type of contract is used when objective contractor performance measures are not practical.
- A Cost-Plus-Incentive-Fee Contract (FAR 16.304) provides reimbursement of the contractor's costs. Additionally, a negotiated fee is paid. The acquirer and contractor negotiate a target cost and a fee schedule. The resulting fee is adjusted based upon the relationship between actual and target costs. This type of contract is used when the level of cost uncertainty is too great for a fixed-price contract. The incentive fee then motivates the contractor to perform efficiently.

- A Cost-Plus-Award-Fee Contract (FAR 16.405-2) provides reimbursement of the contractor's costs, and a base fee, negotiated at the time of contract award. Additionally, through excellent performance (as evaluated by the acquirer), the contractor may earn additional award fees. This type of contract is used when the level of cost uncertainty is too great for a fixed-price contract, and it is impractical to establish pre-defined incentive targets.

The use of incentive fees and the use of award fees are not mutually exclusive, and may be used simultaneously. Contract term may also be used as an incentive, providing extensions or reductions in the term of the contract based upon contractor performance.

4.5.4 Selecting a Contract Type

When selecting a contract type, the acquirer must consider several factors including

- Competition – Competition among responsible suppliers typically results in realistic pricing, enabling the use of fixed-price contracts.
- Complexity – Complex products are inherently more risky than simpler ones. The contract type plays a role in defining the level of risk-sharing between the acquirer and the contractor.
- Acquisition Size – The size of the acquisition as measured by total cost or total effort, has a direct bearing on the choice of a contract type, in that some contract types are not appropriate for larger acquisitions. Additionally, larger acquisitions are inherently more risky than smaller acquisitions. The contract type plays a role in defining the level of risk-sharing between the acquirer and the contractor.
- Requirement Stability – Fixed-price contracts are appropriate only when the product requirements are complete and stable at the time of solicitation. Incomplete requirements or requirements volatility add uncertainty that is unacceptable for fixed-price contracts.
- Cost Estimation – The ability of both the acquirer and the contractor to accurately predict contract costs is crucial to the selection of a contract type. Without accurate cost estimates, fixed-price contracts will not be practical. Cost estimation accuracy derives from a clear definition of requirements and experience with similar systems
- Contractor Capabilities – The contractor's technical capabilities and performance history informs the degree of risk that the acquirer is willing to share. The contractor's financial security influences decisions on cost sharing, payments periods, and more. The sufficiency of the contractor's accounting system is a gating factor for the use of a cost reimbursable contract.
- Acquirer Capabilities – Some contract types are more difficult to administer than others. The ability of the acquirer to address these challenges influences type of contract that is selected.

The strategic choices and drivers of the Contract Approach strategy element are summarized in the strategy profile shown in Table 10.

Table 10: Contract Type Strategy Profile

Strategy Element Range	**Fixed-Price Contracts** • Firm-Fixed-Price (FFP) Contracts • Fixed-Price Contract with Economic Price Adjustment • Fixed-Price Contract with Prospective Price Redetermination • Fixed-Ceiling-Price Contracts with Retroactive Price Redetermination • Firm-Fixed-Price, Level-of-Effort Term Contract **Cost Contracts** • Cost Contract • Cost-Sharing Contract • Cost-Plus-Fixed-Fee Contract **Incentive Contracts** • Fixed-Price Incentive Contract • Fixed-Price Contract With Award Fees • Cost-Plus-Incentive-Fee Contract • Cost-Plus-Award-Fee Contract
Principal driver influences (from Table 6)	**1. Software Criticality** **2. Acquisition Environment Cat. Drivers** a. Policies and Mandates b. Supplier Availability **3. Programmatic Category Drivers** a. Mission Needs and Scope b. Funding *i. Funding Constraints* *ii. Funding Profile* c. Schedule *i. Schedule Constraints* *ii. Urgency* **4. Organizational Category Drivers** a. Program Management Office Capabilities *i. PMO Staff Skills* *ii. PMO Staff Capacity* *iii. PMO Staff Stability* b. Stakeholders *i. Number and Diversity* *ii. Level of Engagement (responsiveness and quality)* *iii. Level of Agreement* c. Supplier Capability *i. Supplier Staff Skills* *ii. Supplier Staff Capacity* *iii. Supplier Staff Stability* *iv. Supplier Performance to Date* **5. Life Cycle Category Drivers** a. Product Definition and Specification *i. Requirements Volatility* *ii. Requirements Understanding* *iv. Interoperability* b. Architecture and Design *i. Precedence* *ii. Quality Attribute Constraints* *iii. Technology Readiness* *iv. Legacy Considerations* *v. COTS / GOTS / Reuse*

4.6 Information Assurance Strategy Element

The DoD defines *information assurance* (IA) as "measures that protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation. This includes providing for the restoration of information systems by incorporating protection, detection, and reaction capabilities [DAU 04].” Systems today are subject to threats throughout their life cycle. The goal of IA and cyber-security in general is to create and maintain a secure system through a combination of secure design techniques, secure operating procedures, and secure support procedures. Security is not a system attribute that is added to a system. It is an attribute that is part of the system definition and impacts nearly all aspects of the system design. Just a few of the security challenges faced by networked systems include:

- Inclusion of malicious code during development and/or support – In either networked or independent systems, developers or maintainers may introduce malicious code during the development or support phase of the system [GAO 04b]. At some future time, or in response to some pre-defined stimulus, such code may cause system failure, alter system operation in some undesirable manner, compromise system data security or integrity, or all of the above.
- Vulnerability to unauthorized access – Systems are often intended for use by a restricted set of users. Without proper security measures (e.g., user authentication, access controls, network monitoring), both networked and independent systems are subject to unauthorized system access.
- Vulnerability to network attacks – Systems are connected to networks to provide access to multiple users at multiple locations. This connection enables the possibility of system attack via the network. A denial-of-service attack can compromise system performance by overwhelming the network with traffic, thereby denying access to the system via the network. Other forms of attack include exploitation of security weaknesses to gain unauthorized access to the system with the intent of unauthorized use, system software destruction, system data extraction, or system data corruption. Such attacks can be targeted at specific systems (i.e., hackers breaking into a target system) or can be more ambiguous (i.e., infection by a network-carried virus, worm, Trojan horse).

IA is a complex task that impacts all phases of a system’s life cycle.[4] Many techniques exist to enhance system security and many strategy elements interact with IA needs. A brief sampling of the IA strategy sub-elements includes:

1. supplier assurance as a means of preventing of malicious code insertion during development or support
2. choice of communication network technology (e.g., open Internet, secure socket layer, Secret Internet Protocol Router Network [SIPRNet])
3. means of authenticating users and maintainers
4. data encryption usage

[4] For additional information, visit the Build Security In Web site at https://buildsecurityin.us-cert.gov/daisy/bsi/home.html.

As an example, the subsections that follow discuss the first of these strategy sub-elements relating to IA.

4.6.1 Supplier Assurance for Prevention of Malicious Code Insertion During Development

One method of reducing the chances of malicious code being inserted into a system during development or maintenance involves careful code inspections during peer reviews to identify unauthorized content, coupled with strict configuration management to ensure that delivered code consists only of known elements. Another method involves analyzing source code and relating it to system requirements to identify the inclusion of unauthorized code (i.e., code that does not directly support the system requirements). Both of these methods are only as effective as the vigilance of the parties tasked with ensuring the code quality; thus, each method depends heavily on the exercise of care in the selection of developers and maintainers. This is the strategy element that we address here.

The software in a newly developed system is often an amalgamation of

- software elements developed by the system developer
- software elements developed by the system developer’s suppliers and/or subcontractors
- software elements reused from previous efforts of the system developer and/or subcontractors
- COTS, GOTS and/or Free/Open Source (F/OS) software elements

Each of these elements introduces its own risks into the system. A strategy for addressing these issues centers on the concept “Be careful who you buy from.” The more secure your source is, the less your product is likely to be affected by this risk. Strategies for understanding and managing the acquisition of a system that is critical to national security might include

1. Use COTS and F/OS products freely
2. Evaluate COTS and F/OS products before incorporation
3. Use COTS and F/OS products only from trusted sources
4. No COTS or F/OS products; all software developed by U.S. sources
5. No COTS or F/OS products; all software developed by cleared U.S. sources

4.6.1.1 Use COTS and F/OS Products Freely

COTS, GOTS, and F/OS software elements pose a unique challenge to security because the developer may have very little insight into the internal workings of these elements. What security measures have been designed into the product? How have they been verified? Has the developer incorporated “back doors” and information-harvesting processes into the product for his own purposes? These are questions that may be unanswerable. As such, liberal use of COTS, GOTS and F/OS products may do little to mitigate information assurance risks.

4.6.1.2 Evaluate COTS and F/OS Products Before Incorporation

It may be possible to evaluate the security features of COTS, GOTS and F/OS components prior to their incorporation into the system. This usually requires the cooperation of the supplier and, at a minimum, access to the component's source code. This strategy primarily involves auditing the security practices claimed by the COTS/GOTS supplier (this may not be possible with F/OS components due to their public nature), evaluating their suitability for the system, and evaluating the consistency of application by the supplier. You can also consider a technical evaluation of the components themselves, although attempting to "test-in" security in this manner rather than designing it into the product is difficult and not highly effective. Another tactic is to examine security advisories, bulletins, and alerts to determine a product's history of vulnerabilities.

4.6.1.3 Use COTS and F/OS Products Only from Trusted Sources

It may be possible to evaluate the trustworthiness of COTS, GOTS, and F/OS component suppliers prior to incorporation of their products into the system. While no widely recognized certification process currently exists, several DoD initiatives are working in this direction. For example, the Office of the Under Secretary of Defense (Acquisition, Technology and Logistics) is presently working with the National Defense Industrial Association Systems Assurance Committee to produce a guidebook on systems assurance that includes elements of supplier assurance. While there is no formal certification for suppliers as yet, it is possible to select individual COTS components that have been certified using standards such as Common Criteria and The Common Criteria Evaluation Validation Scheme.[5]

4.6.1.4 No Use of COTS or F/OS Products; All Software Developed by U.S. Sources

Avoiding COTS, GOTS, and F/OS components eliminates the uncertainty associated with these types of products. Of course, this leaves only the alternative of using developmental items in the construction of the system, a practice that introduces a new set of risks. For software elements developed by the system developer and/or subcontractors, malicious employees may introduce code that compromises system security. Opportunities for personal gain (i.e., blackmail of the developer or user) and revenge in response to an employee grievance are among the common motivations for this activity. If the system is critical to national security, allegiance to a foreign power may also be a motivation. It should also be noted that not all security weaknesses are a result of malice. During development, developers may incorporate various shortcuts around security features to simplify their work. Failure to remove these shortcuts completely from the delivered code may also manifest itself in exploitable security weaknesses. Another important consideration is the development tools used; it may be possi-

[5] For more information, visit the Common Criteria portal at http://www.commoncriteriaportal.org/ and the Common Criteria Evaluation Validation Scheme Web site at http://niap.bahialab.com/cc-scheme/.

ble for developers using commercial tools to unwittingly embed vulnerabilities into the developed code.

The reuse of software elements from previous systems poses similar issues, as the same security weaknesses may plague these elements. Additionally, reused components may include features not intended for use in this system that result in an exploitable security weakness.

4.6.1.5 No Use of COTS or F/OS Products; All Software Developed by Cleared U.S. Sources

Much like the previous strategy choice, this choice focuses on the reliability of the developers. Using only U.S. sources with DoD security clearances provide further reinforcement of this reliability. While the possibility of malicious intent motivated by greed or national disloyalty still exists, the fact that the organization and the staff have been investigated and cleared provides some reduction in this risk.

4.6.2 Selecting a Strategy for Malicious Code Prevention

The strategic choices and drivers of the Supplier Assurance strategy sub-element are summarized in the strategy profile shown in Table 11.

Table 11: Supplier Assurance Strategy Profile

Strategy Element Range	Free use of COTS and F/OS products Evaluate COTS and F/OS products before incorporation Use COTS and F/OS products only from trusted sources No COTS or F/OS products; all software developed by US sources No COTS or F/OS products; all software developed by cleared U.S. sources	
Principal driver influences (from Table 6)	**1. Software Criticality** **2. Acquisition Environment Category Drivers** a. Policies and Mandates b. Supplier Availability **3. Programmatic Category Drivers** a. Mission Needs and Scope b. Funding *i. Funding Constraints* *ii. Funding Profile* c. Schedule *i. Schedule Constraints* *ii. Urgency* **4. Organizational Category Drivers** a. Program Management Office Capabilities *i. PMO Staff Skills* *ii. PMO Staff Capacity* *iii. PMO Staff Stability* b. Stakeholders *i. Number and Diversity* *ii. Level of Engagement (responsiveness and quality)* *iii. Level of Agreement* c. Supplier Capability *i. Supplier Staff Skills* *ii. Supplier Staff Capacity* *iii. Supplier Staff Stability* *iv. Supplier Performance to Date*	**5. Life Cycle Category Drivers** a. Product Definition and Specification *i. Requirements Volatility* *ii. Requirements Understanding* *iii. Quality Attribute Definition Quality* *iv. Interoperability* b. Architecture and Design *i. Precedence* *ii. Quality Attribute Constraints* *iii. Technology Readiness* *iv. Legacy Considerations* *v. COTS / GOTS / Reuse* c. Verification and Test *i. Test Environment Complexity* *ii. Test Environment Availability* *iii. Number of System Configurations* d. Deployment *i. Number of Sites* *ii. User Readiness* *iii. Maintainer Readiness* *iv. Transition / Data Migration* e. Maintenance and Support *i. Number of System Configurations* *ii. Update Readiness* *iii. Support Duration* *iv. Re-Competition Readiness* *v. Operational Environment* *vi. Legacy Considerations* *vii. Complexity of Data Rights* f. Disposal

4.7 Product Support Strategy: Training Strategy Element

Note: Some of the information contained in this section is familiar to most experienced acquisition PMO staff. It is reiterated in this report to lend coherence to the authors' discussion of acquisition strategy elements.

In preparation for a successful product deployment, the acquirer must provide training for personnel engaged with the product including users, installers, operators, and maintainers. The acquirer must also address training needs that will arise later in the program, long after the acquisition program office has been disbanded. Who will train the fifth, or tenth, or one hundredth rotation of operators and maintainers? Addressing this need often results in the creation of a training plan and training materials to "train the trainers."

Regardless of the audience for training, several training strategies are available to the acquirer. Candidate strategies include

- Self-Training – Self-guided tutorials (in either paper or electronic format) provide the trainee with the needed information to perform his or her task. This kind of training, the equivalent of reading the installation manual, user's manual, or service manual for the product, is only appropriate for products that are very easy to install, operate, and maintain. It can be effective in cases where the product installation, operation, and maintenance are highly intuitive and performed by highly capable staff. While this option is usually the least effective type of training, it is also the least costly.
- Computer-Based Training (CBT) – Like self-training, CBT is self-paced. While both occur without an instructor, unlike self-training, CBT provides structure, guidance, and feedback to the trainee. CBT can be designed using proven instructional design methods that establish learning objectives, provide instruction to support those objectives, and verify the trainee's attainment of those objectives. Typically, the CBT system provides instruction in accordance with a defined curriculum using various presentation methods (e.g., text, graphics, animation, audio, video) via a computer's multimedia capabilities. The trainee controls the pace of the presentation, with the ability to repeat and review previously shown information. CBT often includes exercises to provide guided practice and aid the trainee's absorption of knowledge. Throughout the instruction, CBT tests the knowledge of the trainee, provides feedback, and adjusts the remaining path of the instructional sequence to remedy any knowledge deficiencies.

  CBT can be highly effective for all but the most complex products. It is highly economical to deliver courses in this manner; however, course development costs can be very high.
- Distance Learning – When trainees are dispersed over a wide geographic area, distance learning (DL) may be a suitable training option. Although many definitions of distance learning exist, most agree that key attributes include

a. the separation of teacher and learner in space and/or time

b. the volitional control of learning by the student rather than the distant instructor, and

c. noncontiguous communication between student and teacher, mediated by print or some form of technology [Sherry 96]

DL uses technology to bridge the gap created by a lack of immediate and personal communication between instructors and trainees. Like normal classroom instruction, DL is governed by a curriculum, with instructors preparing materials (paper or electronic) for distribution to the trainees. In most cases, the trainees determine the pace of the training. DL includes testing at specific points within the instructional sequence, with the tests evaluated by the instructors who provide feedback and re-direction to the individual trainees as needed. The trainees may maintain periodic bi-directional contact with the instructor via Internet, telephone, video-conference, etc., enabling the trainees to discuss issues and seek clarifications of the instructional materials, and enabling the instructor to better assess the progress of the trainees.

DL can be highly effective for all but the most complex products. Preparation cost is somewhat higher than for classroom training due to the need to adapt training materials for remote access and use. Delivery cost is somewhat higher than typical classroom training due to the infrastructure support activities. On the other hand, a single instructor can typically interact with more distant learning trainees than classroom trainees. The elimination of travel costs for the trainees also provides a significant delivery cost advantage.

- Classroom Training – Unlike self-training and CBT, classroom training is a group activity rather than an individual activity. It is typically instructor-paced, although a good instructor will adapt the pace of the training to the abilities of the majority of his students. Classroom training can be designed using proven instructional design methods that establish learning objectives, provide instruction to support those objectives, and verify the trainee’s attainment of those objectives. Typically, classroom training provides instruction in accordance with a defined curriculum using lectures, discussions, and various presentation aids (e.g. audio, video). Hands on training may be included using either the actual product, or training aids (e.g. simulators, mock-ups). The training often includes exercises to provide guided practice, aiding the trainee’s absorption of knowledge. At specific points within the instructional sequence, the instructor tests the knowledge of the trainee, and provides feedback.

  Classroom training can be highly effective for all types of products. Preparation cost is higher than that for self-learning, but less than that for CBT or distance learning. Delivery cost is significantly higher than self-learning or CBT. While delivery cost may be less than distance learning, total cost, including the travel costs of the students to get to the class site, may be higher. In the DoD, classroom training typically has to be coordinate with a specific training unit. The training unit should be considered a stakeholder in the acquisition.

- Field Training – Field training is accomplished by sending a trained instructor to (or near) the site where the students will use the product. The trainer instructs the students, either individually, or in a group, on the use and/or maintenance of the product. Field training is accomplished in accordance with a defined curriculum. While it may include instruction using lectures and discussions, it is predominantly hands-on training using the actual product. It is typically instructor-paced, rather than student-paced; although a good instructor will adapt the pace of the training to the abilities of the students. Typically, students verify achievement of the learning objectives directly by demonstration of skills to the instructor.

  Field training can be highly effective for all types of products due to the one-on-one interaction with the instructor in the product's intended environment. Preparation cost is often less than classroom training, distance training, or CBT. Delivery cost is significantly higher than all of the alternatives, since it involves transporting the trainer to the training site, and training students individually or, at most, in small groups.

The strategic choices and drivers of the Training strategy element are summarized in the strategy profile shown in Table 12.

Table 12:   Training Strategy Profile

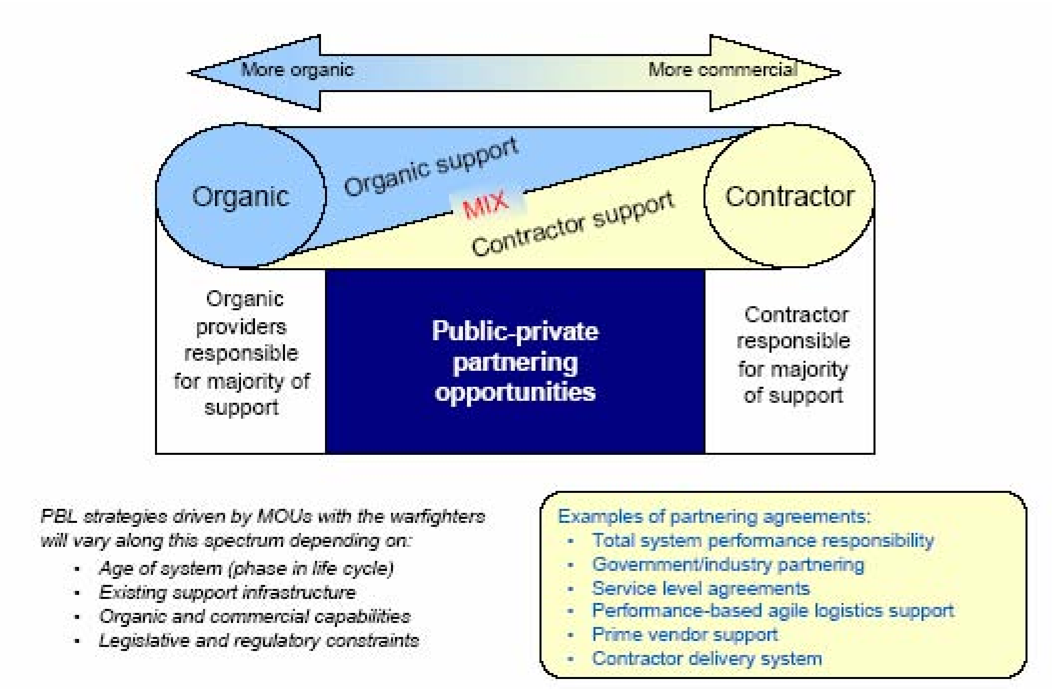
Strategy Element Range	Self-Training Computer-Based Training Distance Learning Classroom Training Field Training
Principal driver influences (from Table 6)	**1. Software Criticality** **2. Acquisition Environment Cat. Drivers** a. Policies and Mandates b. Supplier Availability **3. Programmatic Category Drivers** a. Mission Needs and Scope b. Funding *i. Funding Constraints* *ii. Funding Profile* c. Schedule *i. Schedule Constraints* *ii. Urgency* **4. Organizational Category Drivers** a. Program Management Office Capabilities *i. PMO Staff Skills* *ii. PMO Staff Capacity* *iii. PMO Staff Stability* b. Stakeholders *i. Number and Diversity* *ii. Level of Engagement (responsiveness and quality)* *iii. Level of Agreement* c. Supplier Capability *i. Supplier Staff Skills* *ii. Supplier Staff Capacity* *iii. Supplier Staff Stability* *iv. Supplier Performance to Date* **5. Life Cycle Category Drivers** a. Product Definition and Specification *i. Requirements Volatility* *ii. Requirements Understanding* *iii. Quality Attribute Definition Quality* *iv. Interoperability* b. Architecture and Design *i. Precedence* *ii. Quality Attribute Constraints* *iii. Technology Readiness* *iv. Legacy Considerations* *v. COTS / GOTS / Reuse* d. Deployment *i. Number of Sites* *ii. User Readiness* *iii. Maintainer Readiness* *iv. Transition / Data Migration* e. Maintenance and Support *i. Number of System Configurations* *ii. Update Readiness* *iii. Support Duration* *iv. Re-Competition Readiness* *v. Operational Environment* *vi. Legacy Considerations* *vii. Complexity of Data Rights* f. Disposal

4.8 Product Support Strategy: Source of Support Strategy Element

Note: Some of the information contained in this section is familiar to most experienced acquisition PMO staff. It is reiterated in this report to lend coherence to the authors' discussion of acquisition strategy elements.

After the product is developed, tested, and delivered, the job of the acquisition PMO is finished, but that is not the end of the program. Long after the acquisition PMO has been closed, the product must be maintained. New trainers must be trained to train the new users and maintainers. Newly discovered product defects must be corrected. Product upgrades must be developed and fielded. At the end of its useful life, the product must be disposed of. In short, the product must be supported throughout its remaining life cycle. While this activity often becomes the responsibility of an Operations PMO, it must first be addressed by the acquisition PMO. Failure of the acquisition PMO to adequately address deployment, support, and disposal of the product may make these tasks unreasonably difficult and costly, or even impossible, to perform.

Answering the question "Who will support this product?" is a key factor in the development of a support strategy. Sources of product support range from various forms of organic support (support provided from within the government) to support provided solely by a contractor, as shown in Figure 10 [OUSD AT&L 01].


More organic
More commercial
Organic
Organic support
MIX
Contractor support
Contractor
Organic providers responsible for majority of support
Public-private partnering opportunities
Contractor responsible for majority of support

PBL strategies driven by MOUs with the warfighters will vary along this spectrum depending on:

- *Age of system (phase in life cycle)*
- *Existing support infrastructure*
- *Organic and commercial capabilities*
- *Legislative and regulatory constraints*

Examples of partnering agreements:

- Total system performance responsibility
- Government/industry partnering
- Service level agreements
- Performance-based agile logistics support
- Prime vendor support
- Contractor delivery system

Figure 10: Office of the Under Secretary of Defense for Acquisition, Technology and Logistics' Depiction of the Spectrum of Performance Based Logistics (PBL) Strategies

Candidate support strategies include

- Contractor Logistics Support (CLS) – For CLS, product support is contracted to a commercial contractor. This contractor may be the original developer of the product, an organization that specializes in logistics support, etc. In fact, the support activities may even be divided among multiple contractors; one for field maintenance, another for upgrade development, and yet another for disposal. The role of the contractors is defined in the support contract from the Operations PMO. It includes a definition of the scope of the support effort, as well as requirements for the quality of support (response time, failure rates, mean time to repair, etc.). Cost-plus-incentive-fee or cost-plus-award-fee contracts are commonly used for CLS. Fixed-price contracts are unusual due to the uncertainty of the amount of effort that is required.

  A CLS strategy places demands on the acquirer. The acquirer must ensure sufficient availability and data rights to technical information from the developer to enable the support contractor to perform the maintenance and upgrade tasks. The acquirer must ensure that the reliability and maintainability of the product are adequate to enable the support contractor to fulfill his mission. The acquirer must also ensure that sufficient training is available.

  CLS can be a viable support option for nearly all products. It enables the PMO to draw upon the technical and management strengths of the defense industry to support fielded products. In cases where the acquirer has not negotiated with the developer for sufficient data rights and access to technical information, CLS with the developer as the support contractor may be the ONLY viable strategy. CLS can be an expensive support option; however, with appropriate incentives, the cost can be managed.

- Organic Support – Organic support may take several forms, depending upon which organization within the government provides the support, as shown below:
  - PMO Support – Product support can be provided directly from the Operations PMO by PMO staff trained in the support of the product. Like CLS, a PMO support strategy also places demands upon the acquirer. The acquirer must again ensure sufficient availability and data rights to technical information from the developer to enable the PMO support staff to perform maintenance and upgrade tasks. The acquirer must ensure that the reliability and maintainability of the product are adequate to enable the PMO support staff to fulfill their mission. The acquirer must also ensure that sufficient training is available.

    Although PMO support is not used frequently due to chronic understaffing of PMOs, it can be a viable support option for products of low to medium complexity. PMO support can be a relatively inexpensive support option; however, the quality of support is highly dependent upon the skill and training of the PMO support staff.
  - Indigenous Support – Some products can be supported primarily by the units to which they are fielded. For example, transportation vehicles are often maintained by an on-base maintenance facility (i.e., motor pool) staffed by base staff. Like CLS, an indigenous support strategy places the same demands upon the acquirer. The acquiring PMO must ensure sufficient availability and data rights to technical information

from the developer, that the reliability and maintainability of the product are adequate, and that sufficient training is available.

Indigenous support works effectively on relatively simple products that can be supported without extensive training, and it can be an inexpensive option.

- Depot Support – Product support can be provided by depots or other DoD support facilities.

  Like CLS, a depot support strategy again places the same demands upon the acquirer. The acquiring PMO must ensure sufficient availability and data rights to technical information from the developer, that the reliability and maintainability of the product are adequate, and that sufficient training is available.

  Depot support works effectively on products widely used throughout the DoD. It is applicable to products at all levels of complexity. With sufficient volume of products, the depot can afford the investment needed to establish and maintain a support capability. Additionally, the depot can establish agreements with the product developer to address recurring failures and product upgrades.

Many combinations of organic and contractor support are possible.

The strategic choices and drivers of the Source of Support strategy element are summarized in the strategy profile shown in Table 13.

Table 13: Source of Support Strategy Profile

Strategy Element Range	Contractor Logistics Support PMO Support Depot Support Indigenous Support
Principal driver influences (from Table 6)	**1. Software Criticality** **2. Acquisition Environment Cat. Drivers** a. Policies and Mandates b. Supplier Availability **3. Programmatic Category Drivers** a. Mission Needs and Scope b. Funding *i. Funding Constraints* *ii. Funding Profile* c. Schedule *i. Schedule Constraints* *ii. Urgency* **4. Organizational Category Drivers** a. Program Management Office Capabilities *i. PMO Staff Skills* *ii. PMO Staff Capacity* *iii. PMO Staff Stability* b. Stakeholders *i. Number and Diversity* *ii. Level of Engagement (responsiveness and quality)* *iii. Level of Agreement* c. Supplier Capability *i. Supplier Staff Skills* *ii. Supplier Staff Capacity* *iii. Supplier Staff Stability* *iv. Supplier Performance to Date* **5. Life Cycle Category Drivers** a. Product Definition and Specification *iv. Interoperability* b. Architecture and Design *i. Precedence* *iii. Technology Readiness* *iv. Legacy Considerations* *v. COTS / GOTS / Reuse* d. Deployment *i. Number of Sites* *ii. User Readiness* *iii. Maintainer Readiness* *iv. Transition / Data Migration* e. Maintenance and Support *i. Number of System Configurations* *ii. Update Readiness* *iii. Support Duration* *iv. Re-Competition Readiness* *v. Operational Environment* *vi. Legacy Considerations* *vii. Complexity of Data Rights* f. Disposal

5 Formulating an Acquisition Strategy

5.1 Overview

In Section 3, we defined a set of drivers that influence a program's acquisition strategy and in Section 4, we began to codify an acquisition strategy in terms of strategy elements and corresponding strategic choices.

This section uses the strategy profiles created for the strategy elements and sub-elements (Acquisition Approach, Business Considerations, and Product Support) discussed in Section 4 to illustrate a strategy development method. Note that additional strategy profiles and strategy elements may be addressed in future research.

Many different drivers influence multiple strategy elements and vice versa. Due to the number and the complexity of relationships between the drivers and the strategy elements, it is difficult to determine which strategy choices best fit the drivers. For a strategy that has "M" strategy drivers and "N" strategy elements, the number of relationships could approach (M x N). For example, if there were 10 drivers and 5 strategy elements the number of relationships could approach 50.

Programs need better ways to reason about software risks, formulate acquisition strategies to mitigate software risk, and evaluate their current acquisition strategy's ability to mitigate software risk in an ongoing, systematic manner. In this section, we provide a method for performing a comparative analysis of the drivers, the strategy elements and their corresponding strategic choices.

The method we propose is a graphical, bi-directional method of examining and analyzing the relationships between the drivers and the strategy element strategic choices. The method uses a graphical element, slider bars, to support a more systematic approach to reasoning about software risk. The method is bi-directional in that it enables a program to

1. Analyze strategy drivers and then choose strategies (strategic choices) for each strategy element to minimize the risks induced by those drivers.
2. Choose a strategy (strategic choice) for each strategy element and then analyze the risks induced by the program's drivers.

In this section, we explain

- the strategy development method (Section 5.2)
- the concept of a slider bar (Section 5.3)

- how to diagram a program's software acquisition characteristics, its strategy drivers, its acquisition strategy elements and corresponding strategic choices, and the relationship between the drivers and strategic choices (Section 5.4)

By evaluating strategy drivers and acquisition strategy elements' strategic choices using the slider bar technique, program managers and others can rate their program and system to build an acquisition profile. As a result, implementations will be more sharply focused on mitigating underlying software risks and will exhibit higher levels of acceptance, commitment, and understanding among those who implement and oversee the program's acquisition strategy.

5.2 Strategy Development Method

As discussed in Section 2, factors both within the program and external to the program generate program risks. We call these factors *drivers* and have discussed them at length in Section 3. The goal of an effective acquisition strategy is to addresses the program risks generated by these drivers. No strategy can be sufficiently comprehensive and suitably complex to address all of the risks of a program. However, we can formulate a strategy that addresses the *most significant* risks and leave the remaining risks to be addressed by the risk management process within the program, as shown in Figure 11.

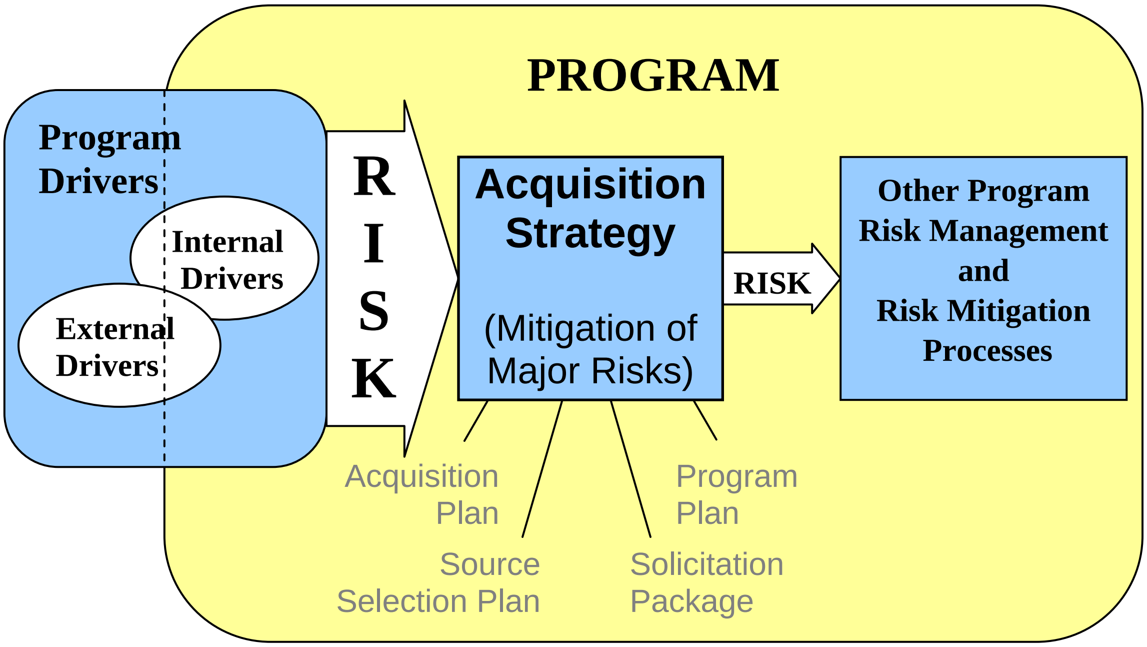


Figure 11: Risk Management Via Acquisition Strategy

To use the acquisition strategy as a means of mitigating program risks, we must

1. identify the program risks
2. identify the elements of the strategy
3. understand the relationship between the risks and the strategy elements' strategic choices

The presumptions of this acquisition strategy development method are that

- The strategy drivers can be related to the production of risk for the program.
- The strategic choices within strategy elements can be ranked in order of their ability to address the program's identified classes of risk.
- The strategic choices within strategy elements can mitigate risks generated by specific strategy drivers.

The next three sections discuss these presumptions in more detail.

5.2.1 Strategy Drivers and Risk

Let's consider the first premise—strategy drivers can be related to the production of risk for the program. For example, "software criticality" is one of the drivers we defined and discussed in Section 3. Clearly, the more critical software is to achieve the required program performance, the more risk it poses to the program. "Requirements understanding," another driver discussed in Section 3, can be considered in the same manner; requirements that are well understood pose less risk to a program than requirements that are not understood.

5.2.2 Strategy Elements and Risk

Let's consider the second premise—strategy choices within strategy elements can be ranked in order of their ability to address the program's identified classes of risk. For the Acquisition Approach strategy element, three strategy choices are suggested in Section 4: single-step, evolutionary/incremental, and evolutionary/spiral. The question that we need to address is "How do these three choices compare in terms of mitigating program risk?" To answer this question, one line of reasoning is as follows:

- The evolutionary/spiral acquisition approach was specifically developed as a means of addressing program risk. It enables the acquirer to perform a risk-driven partitioning of the program into sequential acquisition spirals. Each spiral is defined to address the most significant risk issues of the program at that phase of its development. In this manner, the acquirer can address and perform early mitigation of the highest risks to the program. This acquisition approach is particularly good at dealing with incomplete and poorly defined requirements in that early spirals can provide useful demonstrations of capabilities to the stakeholders, enabling them to better understand the direction of the development, and clarify their statements of need.
- The evolutionary/incremental acquisition approach is also a means of addressing program risk. It enables deferment of some portions of the development to later times in the program to permit time for technology development, staff development, and so on. It is not particularly suitable for dealing with incomplete and poorly defined requirements in that all planned increments are based upon needs defined at the start of the program.
- The single-step acquisition approach is least capable of addressing program risk largely because it assumes a low-risk program to begin with. It requires the development of a

start-to-finish execution plan based on needs defined at the start of the program. As such, it is least suitable for dealing with incomplete and poorly defined requirements.

In light of this discussion, we could rank the strategy choices as follows:

Acquisition Approach Strategy	**Risk Mitigation Capabilities**
Single-Step	Low
Evolutionary/Incremental	Medium
Evolutionary/Spiral	High

As a second example, let us look at the Business Considerations: Competition strategy element. Section 4 provides three candidate strategy choices: Full and Open Competition, Full and Open Competition After Exclusion of Sources, and Sole Source Contracting.

We could rank the risk mitigation capabilities of these strategy choices, according to the following reasoning:

- A Full and Open Competition strategy provides the greatest degree of mitigation for risks to achieving desired performance. It mitigates performance risk by gathering technical and management approaches from the widest population of bidders. It mitigates cost and schedule risk by placing all bidders in a competitive environment encouraging them to plan and execute with optimum efficiency.
- A Full and Open Competition After Exclusion of Sources strategy is less robust at mitigating program risk. Capable sources are excluded based upon goals for Small Business utilization, goals for disadvantaged business utilization, etc. Because the population of respondents is reduced, mitigation of performance risk is also reduced. Because the competitive environment is limited, cost and schedule risk mitigation is likewise limited.
- A Sole Source Contracting strategy offers the least degree of risk mitigation. Only one technical approach from one supplier is considered. No competition is present to encourage efficient operation and mitigate cost and schedule risk.

In light of this discussion, we could rank the strategy choices as follows:

Competition Strategy	**Risk Mitigation Capabilities**
Sole Source Contracting	Low
Full and Open Competition after Exclusion of Sources	Medium
Full and Open Competition	High

The examples above are of a notional nature. Other circumstances can be envisioned that would lead to different orderings when ranking the strategy choices. As such, the ranking process is highly contextual and requires careful consideration of the circumstances of each project.

5.2.3 Relating Strategy Drivers to Strategy Elements

From the preceding discussion, it’s clear that strategy drivers influence program risk. We can also observe how some strategic choices are better able to address risk than others. Relating both strategy drivers and strategy elements to risk provides a means for correlating drivers and strategic choices; programs with higher risks, as evidenced by the strategy drivers, should employ strategic choices better able to address program risks. However, to accomplish this, we must first know which risks can be mitigated by strategy element strategic choices.

As noted in Section 5.2.2, these linkages will differ for different programs. As such, the linkages must be carefully evaluated in the context of each individual program. Although a tedious process, we used a group brainstorming method to understand these relationships. Addressing each strategy element individually, we first reviewed the range of strategy choices determined for that element. Subsequently, we discussed the impact of each individual strategy driver upon the strategy element. Much of the discussion consisted of postulating program scenarios to expose the relationships between the driver and the strategy element.

Table 14 and Table 15 provide a notional summary of this relationship for a software-intensive system. The linkages between the strategy drivers and strategy elements are characterized notionally as

- Strong (S) – A significant correlation exists between the risk posed by the strategy driver and the risk mitigation capabilities of the strategy element.
- Medium (M) – A moderate correlation exists between the risk posed by the strategy driver and the risk mitigation capabilities of the strategy element.
- Weak or None (no entry) – A small correlation or no correlation exists between the risk posed by the strategy driver and the risk mitigation capabilities of the strategy element.

Table 14: Strategy Driver – Strategy Element Mapping

KEY

	Weak or No Linkage
M	Medium Linkage
S	Strong Linkage

		STRATEGY ELEMENTS												
		Pgrm Structure		**Acquisition Approach**	**Business Considerations**				**Risk Management**	**Information Assurance-Development Source**	**Test and Evaluation**	**Product Support Strategy**		
		Milestone Decision Points	Acquisition Phases		Competition	Solicitation	Source Selection	Contract Approach				Training	Installation	Source of Support
STRATEGY DRIVERS	**Software Criticality**	**S**	**S**	**S**	**S**	**S**	**S**	**S**		**S**	**S**	**S**	**S**	**M**
	Acquisition Environment													
	Policies and Mandates	**S**	**S**	**S**	**S**	**S**	**S**	**S**		**S**	**S**	**S**	**S**	**S**
	Supplier Availability	**S**	**S**	**S**	**S**	**S**	**S**	**S**		**S**		**M**	**M**	**S**
	Programmatic Category Drivers													
	Mission Needs and Scope	**S**	**S**	**S**	**S**	**S**	**S**	**S**	**M**	**S**	**S**	**S**	**S**	**S**
	Funding													
	Funding Constraints	**S**	**S**	**S**	**S**	**S**	**S**	**S**	**M**	**S**	**S**	**S**	**S**	**S**
	Funding Profile	**S**	**S**	**S**	**S**		**M**	**S**		**M**	**M**	**M**	**M**	**S**
	Schedule													
	Schedule Constraints	**S**	**S**	**S**	**S**	**S**	**S**	**S**	**M**	**M**	**S**	**S**	**S**	**S**
	Urgency	**S**	**S**	**S**	**S**	**S**	**S**	**S**	**M**	**M**	**S**	**S**	**S**	**S**
	Organizational Category Drivers													
	Program Management Office Capabilities													
	PMO Staff Skills	**M**	**M**	**M**	**M**	**M**	**S**	**M**	**S**	**M**	**M**	**M**	**M**	**S**
	PMO Staff Capacity	**M**	**M**	**M**	**M**	**M**	**S**	**M**	**S**	**M**	**M**	**M**	**M**	**S**
	PMO Staff Stability	**M**	**M**	**M**	**M**	**M**	**S**	**M**	**S**	**M**	**M**	**M**	**M**	**S**
	PMO Process Focus						**S**		**S**		**S**			
	Stakeholders													
	Number and Diversity	**S**		**S**			**S**	**S**		**M**	**M**	**S**	**S**	**S**
	Level of Engagement	**S**		**S**			**S**	**S**		**M**	**M**	**S**	**S**	**S**
	Level of Agreement	**S**		**S**			**S**	**S**		**M**	**M**	**S**	**S**	**S**
	Supplier Capability													
	Supplier Staff Skills						**S**	**S**	**S**	**S**	**S**	**S**	**S**	**S**
	Supplier Staff Capacity						**S**	**S**	**S**	**S**	**S**	**S**	**S**	**S**
	Supplier Staff Stability						**S**	**S**	**S**	**S**	**S**	**S**	**S**	**S**
	Supplier Performance to Date						**S**	**S**	**S**	**S**	**S**	**S**	**S**	**S**

Table 15: Strategy Driver – Strategy Element Mapping (continued)

KEY

	Weak or No Linkage
M	Medium Linkage
S	Strong Linkage

		STRATEGY ELEMENTS												
		Prog-ram Struc-ture			**Business Considerations**							**Product Support Strategy**		
		Milestone Decision Points	Acquisition Phases	**Acquisition Approach**	Competition	Solicitation	Source Selection	Contract Approach	**Risk Management**	**Information Assurance – Dev't Source**	**Test and Evaluation**	Training	Installation	Source of Support
STRATEGY DRIVERS	**Life-Cycle Category Drivers**													
	Product Definition and Specification													
	Requirements Volatility	S	S	S	S	S	S	S	S	S	M	M	M	
	Requirements Understanding	S	S	S	S	S	S	S	S	S	M	M	M	
	Quality Attribute Definition Quality	S	S	S	S	S	S		M	S	S	M		
	Interoperability	S	S	S	S	S	S	S	S	S	S	S	S	S
	Architecture and Design													
	Precedence	S	S	S	S	S	S	S	S	S	S	S	S	S
	Quality Attribute Constraints	S	S	S	S	S	S	S	M	S	S	M		
	Technology Readiness	S	S	S	S	S	S	S	S	S	S	S	S	S
	Legacy Considerations	S	S	S	S	S	S	S	S	S	S	S	S	S
	COTS / GOTS / Reuse	S	S	S	S	S	S	S	S	S	S	S	S	S
	Verification and Test													
	Test Environment Complexity								M	M	S			
	Test Environment Availability								M	M	S			
	Number of System Configurations								M	M	S			
	Deployment													
	Number of Sites								M	M	S	S	S	S
	User Readiness								M	M	S	S	S	S
	Maintainer Readiness								M	M	S	S	S	S
	Transition / Data Migration								M	M	S	S	S	S
	Maintenance and Support													
	Number of System Configurations									M	S	S	S	S
	Update Frequency									M	S	S	S	S
	Support Duration									M	S	S	S	S
	Re-Competition Readiness									M		M		S
	Operational Environment									M	S	S	S	S
	Legacy Considerations									M	M	M	S	S
	Complexity of Data Rights									M		S	S	S
	Disposal		M							S		S		M

5.3 Slider Bar Technique

Slider bars are a graphical way to visualize a range or continuum of values. The conceptual elements of a slider bar are

- a label defining the domain of the slider bar
- endpoints defining the range of the slider bar
- a “slider” that indicates the current value within the range defined by the endpoints

For example, if we were to represent the speaker volume of a car stereo, the endpoints might be “soft” and “loud,” as shown in Figure 12. The “slider” shows you where the current value of the volume is on the continuum (about half-volume).

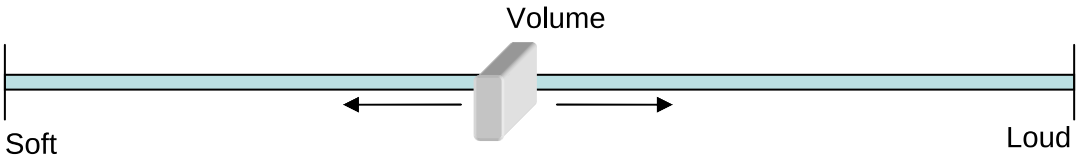


Figure 12: Stereo Slider Bar Example

5.4 Using Slider Bars to Profile Strategy Drivers, Acquisition Strategies, and Their Relationships

Slider bars can be used to visually represent program strategy drivers and many acquisition strategies. They offer the ability to relate strategy drivers to strategy choices, using program risk as a common denominator. Slider bars can be used to graphically visualize a program’s software acquisition profile and frame a discussion about a program’s strategy drivers, risks, and acquisition strategy. After the program’s strategy drivers have been profiled, the slider bars can be used to reason about the efficacy of a given acquisition approach and its ability to address the program’s software risks.

This section introduces readers to the process step-by-step. The first two sections provide some of the basic steps, which build up to more complex uses that are explained in the last two sections.

- Section 5.4.1 explains how to use slider bars to represent a program’s software drivers.
- Section 5.4.2 explains how to use slider bars to represent acquisition strategies that have a range of values.
- Section 5.4.3 explains how to evaluate strategy drivers to help formulate an acquisition strategy. It offers a seven-step approach to help guide acquisition strategy development.
- Section 5.4.4 discusses an approach to help evaluate an existing strategy given the program’s current software acquisition profile.

Programs need better ways to reason about software risks, formulate acquisition strategies to mitigate software risk, and evaluate the ability of current acquisition strategies to mitigate software risk in an ongoing, disciplined manner. By using the slider bar technique to represent its software acquisition profile, a program can gain valuable insight into its software risk.

5.4.1 Diagramming Strategy Drivers

Section 3 described the software risk categories and drivers in detail. This section explains how to diagram strategy drivers using slider bars. Each driver can be represented as a continuum on a slider bar. Each slider bar consists of two extremes of behavior that characterize the driver.

Figure 13 displays a template for visually representing a driver using a continuum bar with appropriate annotations.

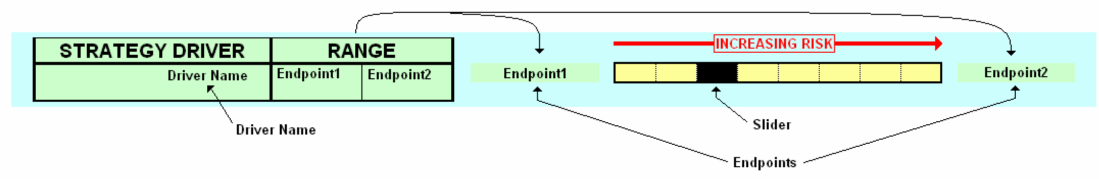


Figure 13: Slider Bar Template

The “Driver Name” is the name of the program driver being evaluated. The endpoints delineate the continuum for the driver. For example, if the driver is “Software Criticality” the endpoints could be “Very Low” and “Very High,” as shown in Figure 14.

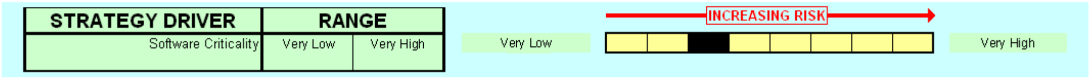


Figure 14: Software Criticality Slider Bar

Next, the program manager or other program personnel identify and mark a point on each slider that best represents the program’s software risk exposure for that element. The person or people making the selection should be able to defend why the particular location on the slider bar was selected. The selected point on a slider bar denotes the assessment of the driver by an individual or a group. This technique is subjective. It is not meant to provide a definitive and quantified assessment of program risk; rather, it is meant to be used as a tool to spur discussion and to clarify thought. For example, a program manager might assess her program at a different point on the slider bar than her manager or direct-reports. This could result in a discussion about what the true status or risk is to the program regarding that driver, leading to a better understanding of the risk.

As a further example of using slider bars, we will consider three acquisition projects:

1. Project 1 is the acquisition of a replacement Jeep. The product contains very little software. The requirements are well known and major changes are not expected.
2. Project 2 is the acquisition of a new armored vehicle. Similar, but not identical, vehicles have been acquired previously. The product contains a moderate amount of software and a few new software requirements, which are not fully defined, are expected.
3. Project 3 is the acquisition of a new unmanned autonomous ground vehicle (UAGV). No vehicles with the required capabilities exist and the technology needed to produce the vehicle is not fully developed. The vehicle requires a moderate amount of software. The system requirements are not well defined; therefore the software requirements are not defined (or can't be defined).

Figure 15 shows how our assessment of requirements volatility for these three projects early in the acquisition might be marked. As time progresses and the new software requirements become better defined, the assessment could move to the left on the slider bar during the iterative process of profiling the program and evaluating the acquisition strategy.

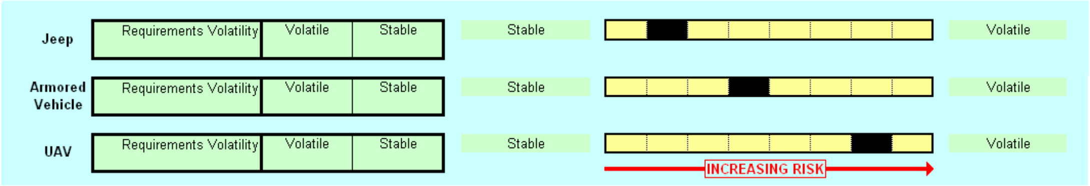


Figure 15: Vehicle Acquisition Example

Appendix A provides a hard-copy template for profiling the strategy drivers discussed in Section 3. In addition, the ASDT is available for download on the SEI Publications Web site at http://www.sei.cmu.edu/publications/publications.html. The ASDT is a Microsoft Excel-based workbook that supports a more automated approach for profiling strategy drivers.

5.4.2 Diagramming Acquisition Strategy Elements

Slider bars can also be used to graphically represent different strategic choices for a specific strategy element. Slider bars cannot be used to diagram all strategy elements, but they can be used to visualize strategies that have a risk mitigation gradation associated with them. For many acquisition strategy elements, the strategic choices can be ranked based upon their risk mitigation capabilities, as discussed in Section 5.2.2. These strategy elements can be represented by slider bars in a manner similar to that of the strategy drivers

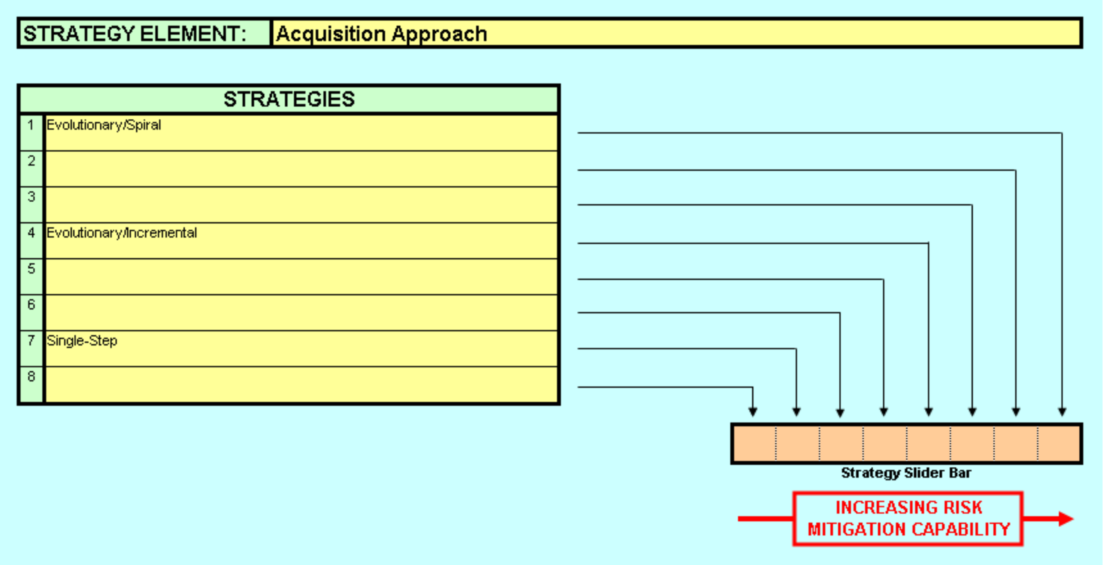
Section 4 described the acquisition strategy elements at a high level. This section explains how to diagram acquisition strategic choices for a particular strategy element using slider bars.

As examples, consider the Acquisition Approach and Business Considerations: Competition strategy elements and their associated risk mitigation capabilities, as discussed in Section 5.2.2:

Acquisition Approach Strategy	**Risk Mitigation Capabilities**
Single-Step	Low
Evolutionary/Incremental	Medium
Evolutionary/Spiral	High

Competition Strategy	**Risk Mitigation Capabilities**
Sole Source Contracting	Low
Full and Open Competition after Exclusion of Sources	Medium
Full and Open Competition	High

These strategy element strategic choices can be represented by slider bars, as shown in Figure 16 and Figure 17.



STRATEGY ELEMENT:	Acquisition Approach

STRATEGIES	
1	Evolutionary/Spiral
2	
3	
4	Evolutionary/Incremental
5	
6	
7	Single-Step
8	

Figure 16: Slider Bar for Acquisition Approach Strategy Element

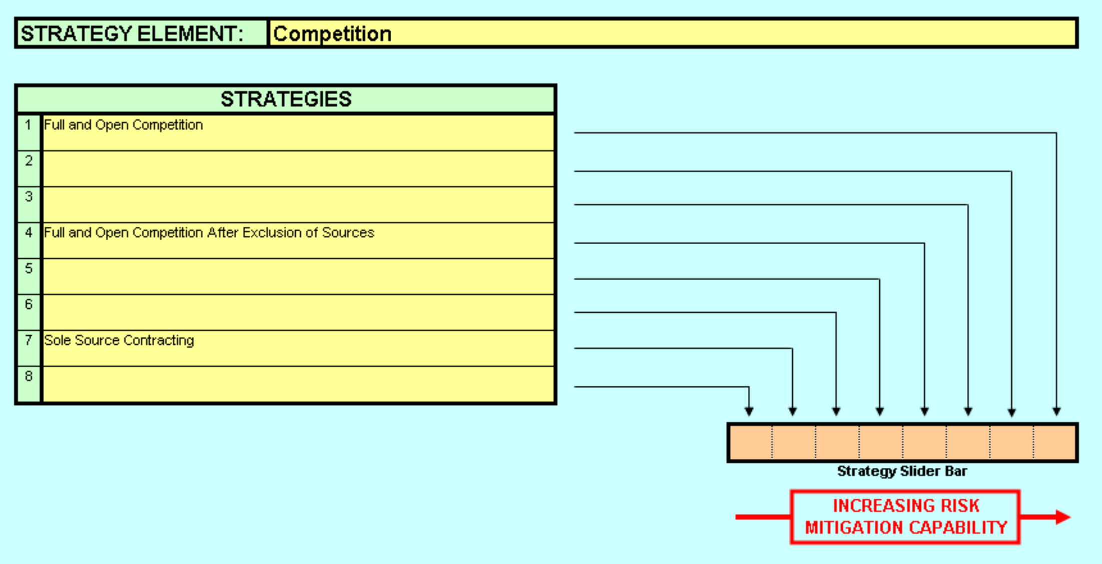

STRATEGY ELEMENT:
Competition
STRATEGIES
1 Full and Open Competition
2
3
4 Full and Open Competition After Exclusion of Sources
5
6
7 Sole Source Contracting
8
Strategy Slider Bar
INCREASING RISK MITIGATION CAPABILITY

Figure 17: Slider Bar for Competition Strategy Element

5.4.3 Developing a Strategy by Evaluating Strategy Drivers

By relating strategy drivers, strategy elements, and corresponding strategic choices to risk, as illustrated in the previous two sections, we have established a basis for correlating them. When developing an acquisition strategy, the process of mapping strategy drivers to strategy element strategic choices consists of nine steps:

1. Define the objectives of the acquisition.
2. Identify and evaluate the factors that drive the program.
3. Decompose the strategy into individual strategy elements.
4. For the selected strategy element, identify the potential strategic choices, and rank them in order of their risk mitigation capabilities for your particular program.
5. Evaluate the strategy drivers for the program to identify those that influence the strategy element through the introduction of risk that may be mitigated by strategy element.
6. Define the relationship between the risk generated by the strategy driver and the risk mitigation capabilities of the strategy element.
7. Map the driver evaluations from Step 1 to the strategy element using the slider bars.
8. Choose the strategy that best mitigates the high-risk elements.
9. Identify residual risks.

The next sections describe each of these steps in more detail, using the Business Considerations: Competition strategy element as an example.

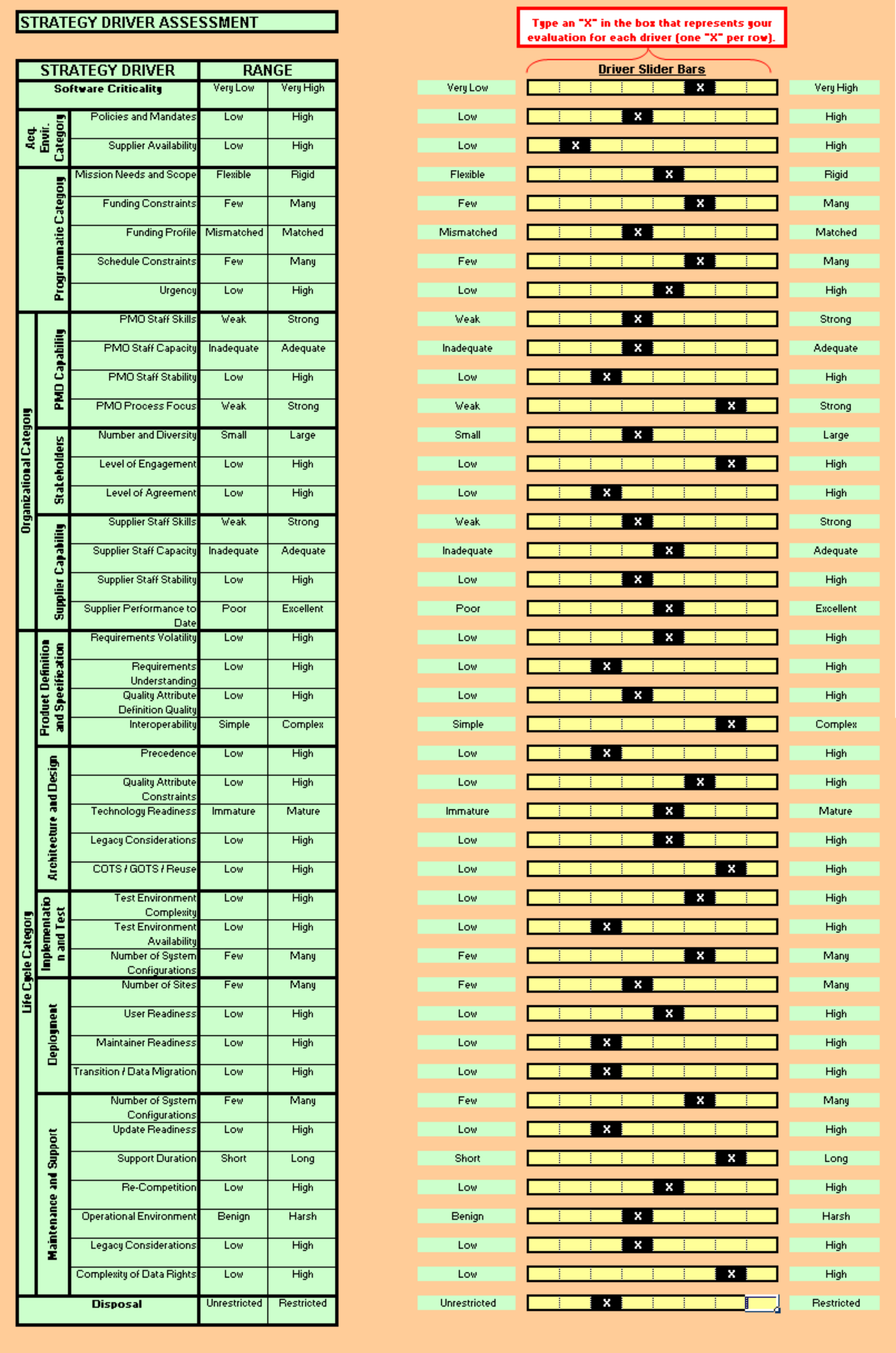
5.4.3.1 Step 1 – Define Acquisition Objectives

Start with a clear definition of the objectives of the acquisition. For example, in the DoD, these objectives are based on the initial program inputs originating with the Joint Capabilities Integration and Development System (JCIDS) and Joint Requirements Oversight Council (JROC). These objectives are also influenced by inputs from end users and other program stakeholders. The acquisition objectives should be clearly documented and kept up-to-date throughout the acquisition life cycle. These objectives form the baseline on which the acquisition and the product development will proceed.

5.4.3.2 Step 2 – Identify and Evaluate Drivers

Now evaluate each of the chosen strategy drivers for your program. In essence, you are making a judgment about the conditions of your program in the context of the range of values defined for the strategy driver. As noted earlier, this is a subjective process intended to spur discussion and to clarify thought; it is not meant to provide a definitive and quantified assessment of program risk.

One effective method of evaluating drivers is to do so as a group within the program office. For example, relevant program office staff (the PM, deputy PM, Chief Engineer, Contracting Officer, and others) meet to discuss the program and the strategy drivers influencing it. Each then completes an independent evaluation of all of the strategy drivers. Subsequently, the group reconvenes and reconciles the independently generated evaluations. Each strategy driver is reviewed and discussed to reach a consensus evaluation. The drivers can then be marked on the slider bars, as shown in Figure 18.



STRATEGY DRIVER ASSESSMENT

Type an "X" in the box that represents your evaluation for each driver (one "X" per row).

		STRATEGY DRIVER	RANGE			Driver Slider Bars								
		Software Criticality	Very Low	Very High	Very Low						X			Very High
Acq. Envir. Category		Policies and Mandates	Low	High	Low				X					High
		Supplier Availability	Low	High	Low		X							High
Programmatic Category		Mission Needs and Scope	Flexible	Rigid	Flexible					X				Rigid
		Funding Constraints	Few	Many	Few						X			Many
		Funding Profile	Mismatched	Matched	Mismatched				X					Matched
		Schedule Constraints	Few	Many	Few						X			Many
		Urgency	Low	High	Low					X				High
Organizational Category	PMO Capability	PMO Staff Skills	Weak	Strong	Weak				X					Strong
		PMO Staff Capacity	Inadequate	Adequate	Inadequate				X					Adequate
		PMO Staff Stability	Low	High	Low			X						High
		PMO Process Focus	Weak	Strong	Weak							X		Strong
	Stakeholders	Number and Diversity	Small	Large	Small				X					Large
		Level of Engagement	Low	High	Low							X		High
		Level of Agreement	Low	High	Low			X						High
	Supplier Capability	Supplier Staff Skills	Weak	Strong	Weak				X					Strong
		Supplier Staff Capacity	Inadequate	Adequate	Inadequate					X				Adequate
		Supplier Staff Stability	Low	High	Low				X					High
		Supplier Performance to Date	Poor	Excellent	Poor					X				Excellent
Life Cycle Category	Product Definition and Specification	Requirements Volatility	Low	High	Low					X				High
		Requirements Understanding	Low	High	Low			X						High
		Quality Attribute Definition Quality	Low	High	Low				X					High
		Interoperability	Simple	Complex	Simple							X		Complex
	Architecture and Design	Precedence	Low	High	Low			X						High
		Quality Attribute Constraints	Low	High	Low						X			High
		Technology Readiness	Immature	Mature	Immature					X				Mature
		Legacy Considerations	Low	High	Low					X				High
		COTS / GOTS / Reuse	Low	High	Low							X		High
	Implementatio n and Test	Test Environment Complexity	Low	High	Low						X			High
		Test Environment Availability	Low	High	Low			X						High
		Number of System Configurations	Few	Many	Few						X			Many
	Deployment	Number of Sites	Few	Many	Few				X					Many
		User Readiness	Low	High	Low					X				High
		Maintainer Readiness	Low	High	Low			X						High
		Transition / Data Migration	Low	High	Low			X						High
	Maintenance and Support	Number of System Configurations	Few	Many	Few						X			Many
		Update Readiness	Low	High	Low			X						High
		Support Duration	Short	Long	Short							X		Long
		Re-Competition	Low	High	Low					X				High
		Operational Environment	Benign	Harsh	Benign				X					Harsh
		Legacy Considerations	Low	High	Low				X					High
		Complexity of Data Rights	Low	High	Low							X		High
		Disposal	Unrestricted	Restricted	Unrestricted			X						Restricted

Figure 18: Evaluated Strategy Drivers

5.4.3.3 Step 3 – Decompose Strategy into Elements

As noted in Section 4.1, an acquisition strategy comprises a number of individual strategy elements. The acquisition planner must identify the applicable strategy elements based on the scope of the acquisition. The result is a list of strategy elements similar to that in Table 5.

5.4.3.4 Step 4 – Rank Strategic Choices by Risk

In Section 5.2.2 we identified three strategy choices for the Business Considerations: Competition strategy element and ranked them in terms of risk mitigation capabilities as follows:

Competition Strategy	Risk Mitigation Capabilities
Sole Source Contracting	Low
Full and Open Competition after Exclusion of Sources	Medium
Full and Open Competition	High

The premise of this ranking was that broader and more open competition elicits a wider range of ideas and options for the acquirer to choose from, thereby providing greater opportunities for risk reduction. There are some conditions under which this premise may not hold, so be sure to evaluate the rankings for your particular program.

5.4.3.5 Step 5 – Identify Strategy Drivers

In the notional example in Table 14, we find that the Business Considerations: Competition strategy element is influenced by the following strategy drivers:

Strategy Driver	Level of Influence
Software Criticality	Strong (S)
Policies and Mandates	Strong (S)
Supplier Availability	Strong (S)
Mission Needs and Scope	Strong (S)
Funding Constraints	Strong (S)
Funding Profile	Strong (S)
Schedule Constraints	Strong (S)
Urgency	Strong (S)
PMO Staff Skills	Medium (M)
PMO Staff Capacity	Medium (M)
PMO Staff Stability	Medium (M)
Requirements Volatility	Strong (S)
Requirements Understanding	Strong (S)
Quality Attribute Definition Quality	Strong (S)
Interoperability	Strong (S)

Precedence	Strong (S)
Quality Attribute Constraints	Strong (S)
Technology Readiness	Strong (S)
Legacy Considerations	Strong (S)
COTS / GOTS / Reuse	Strong (S)

While Table 14 identifies strategy driver/strategy element mappings for a typical program, your program may differ. It is advisable to review all of the strategy drivers to see if they influence the chosen strategy element for your program.

5.4.3.6 Step 6 – Relate Drivers to the Strategy Element

Next, analyze each of the drivers in terms of the risks generated that can be mitigated by a competitive strategy selection. The following paragraphs represent such an analysis based on the Business Considerations: Competition strategy element example.

Software Criticality – Software is typically a risky part of system development. As such, the more critical the software is to the program, the more risk it imparts to the program. Elicitation of more ideas through wider competition provides some mitigation of this risk.

Higher Software Criticality ⇒ Higher Project Risk

Policies and Mandates – Policies and mandates applied to a program restrict the latitude of program decisions and often influence the technical development of the program, potentially forcing it into non-optimal solutions. More competition may provide different methods for addressing some of the constraints imposed by policies and mandates.

More Policies and Mandates ⇒ Higher Project Risk

Supplier Availability – A lack of qualified suppliers impacts the competitive strategy. Without several capable suppliers, full and open competition may not be possible. Furthermore, attempting full and open competition in the absence of qualified suppliers simply invites bids by unqualified suppliers; an occurrence that places additional demands on the program office during source selection, but provides no real benefit to the program. Alternatively, the presence of a number of qualified suppliers enables competition. Competition reduces performance risk by eliciting more ideas and more alternative solutions. Competition also reduces cost and schedule risk by encouraging optimum performance from the suppliers.

Fewer Suppliers ⇒ Higher Project Risk

Mission Needs and Scope – Stringent, non-negotiable mission needs limit the flexibility of the PMO and constrain the trades and technical alternatives that can be considered. This drives the demands that the PMO clearly specify the required performance in unambiguous and complete terms and ensure that the supplier delivers exactly that level of performance. In choosing a supplier, the PMO must assess the product performance risks of the suppliers proposed approach to ensure that the required performance will be met. Broader competition helps the PMO accomplish this by providing a number of alternatives to contrast and compare in the search for the optimal solution.

Likewise, programs of greater scope challenge the PMO. Larger scope implies a greater challenge for the supplier, making the PMO's choice of the "best" supplier more important. Again, broader competition helps the PMO find the supplier best able to handle the challenges of a large program.

More Stringent Mission Needs ⇒ Higher Project Risk
Larger Project Scope ⇒ Higher Project Risk

Funding Constraints – Rigid funding constraints may force the acquirer to choose less capable suppliers, to reduce program office staff to marginal levels, to choose sub-optimal technical solutions, and so on. Elicitation of more ideas through wider competition helps the acquirer to choose the most capable supplier and the best technical solution within the funding constraints.

Rigid Funding Constraints ⇒ Higher Project Risk

Funding Profile – It is important that the funding profile match the needs of the program. Inadequate funding in early phases of the program may force delays of critical aspects of the development effort such as requirements development, systems engineering, trade studies, and so on. Elicitation of more alternatives through wider competition helps the acquirer to choose the supplier with the best approach for addressing funding limitations. Additionally, competition in general tends to drive prices down.

Mismatched Funding Profile ⇒ Higher Project Risk

Schedule Constraints – Rigid schedule constraints can have negative impacts on all phases of a program. Such constraints may limit the acquirer's ability to produce a comprehensive and accurate solicitation, forcing the acquirer to "shortcut" critical early activities such as requirements analysis and systems engineering. The acquirer may also be forced to reduce the amount of time devoted to source selection, which could result in the choice of less capable suppliers and sub-optimal technical solutions. Schedule constraints imposed on the supplier may have equally dire consequences by forcing the supplier to abandon proven processes, take shortcuts in analysis and design, choose sub-optimal solutions, and perform marginal or inadequate verification.

The impact of schedule constraints on a competition strategy can be viewed in two opposing ways. When faced with a tight schedule, the acquirer may focus on reducing solicitation and award time in order to get a supplier on contract sooner. This creates a tendency toward using sole source contracting to eliminate the time needed for proposal evaluation and source selection. This may save some time; however, it commits the program to the approach and the schedule of just one supplier. Alternatively, broader competition may require more time to get to contract award. However, if the need for rapid delivery is stressed in the solicitation and is a factor in the contract award, the competing bidders will strive for the shortest delivery. The time saved in the execution phase of the program often outweighs the time saved in the pre-award phase of the program.

Rigid Schedule Constraints ⇒ Higher Project Risk

Urgency – An urgent need for the program deliverable generates schedule pressure. The PMO must find a supplier not only capable of producing the required performance, but also capable of producing it quickly enough. When choosing a supplier, the PMO must assess the ability of the bidders to meet the required schedule. Broader competition helps the PMO accomplish this by providing a number of alternatives to contrast and compare in the search for the supplier most capable of meeting the required delivery deadlines.

High Urgency ⇒ Higher Project Risk

PMO Staff Skills – A less skillful PMO staff may face challenges managing stakeholder involvement, developing a high-quality solicitation, evaluating contractor proposals, choosing the best supplier, monitoring and controlling supplier performance, and so on. All of these issues increase program risk. The interaction of this driver with the formation of a competitive strategy is problematic. To assess this interaction, we will evaluate the impact of this driver on several competitive strategies

- Full and Open Competition – A less skilled PMO may faces challenges in the creation of an RFP and SOW, in source selection, and in contract negotiations. Further challenges in monitoring and controlling the selected contractor and negotiating changes in cost, scope, and schedule also follow.
- Sole Source Contracting – A less skilled PMO may face challenges in the creation of a SOW, in identifying the appropriate source, and in negotiating a contract. Further challenges in monitoring and controlling the selected contractor and negotiating changes in cost, scope, and schedule also follow.

PMOs face similar challenges with these two strategies; both include challenges in SOW creation, contract negotiations, contractor monitoring and control, and change negotiation. The primary difference is in the solicitation and source selection process.

- For full and open competition, the PMO must generate an RFP, evaluate proposals, and choose a supplier. This process is defined in considerable detail in the FARs.
- For sole source contracting, the PMO must identify and select a supplier. This process is also defined in considerable detail in the FARs.

The execution of a full and open competition contract is probably more challenging for a PMO than the execution of a sole source contract. On the other hand, the success of the program rests heavily on finding a capable and committed supplier; a result more likely to be attained with full and open competition. Selecting the "right" supplier probably poses a greater risk than the additional challenges of full and open competition during solicitation and source selection. Furthermore, these solicitation and source selection challenges may be mitigated by temporarily augmenting the PMO staff with Systems Engineering and Technical Assistance (SETA) contractors and outside support services. The net result of this analysis is a determination that a less skilled PMO staff encourages the use of broader competition.

Lower PMO Staff Skills ⇒ Higher Project Risk

PMO Staff Capacity – Inadequate PMO staff capacity has much the same impact as inadequate PMO staff skills. Through a similar analysis, we can reach a similar conclusion that inadequate PMO staff capacity encourages the use of broader competition.

Inadequate PMO Staff Capacity ⇒ Higher Project Risk

PMO Staff Stability – Inadequate PMO staff stability has much the same impact as inadequate PMO staff skills and inadequate staff capacity. Through a similar analysis, we can reach a similar conclusion that inadequate PMO staff stability encourages the use of broader competition.

Inadequate PMO Staff Capacity ⇒ Higher Project Risk

Requirements Volatility – Poorly defined and volatile requirements introduce program risk. Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the most flexible technical solution, providing some mitigation of this risk.

High Requirements Volatility ⇒ Higher Project Risk

Requirements Understanding – Like high requirements volatility, poorly understood requirements introduce program risk. Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the most flexible technical solution, providing some mitigation of this risk.

Poor Requirements Understanding ⇒ Higher Project Risk

Quality Attribute Definition Quality – Poor quality attribute (QA) definitions (definitions for non-functional requirements such as reliability, supportability, security, and so on) introduces risk to both the PMO and the supplier. Unreasonable QA definitions introduce cost risk, schedule risk, and the risk that these requirements may not be achieved. Absent QA definitions introduce the risk that the product may not meet the needs of the users and maintainers. The challenge of developing a product that meets poorly defined quality attributes requires a highly capable supplier. Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the best technical solution, providing some mitigation of this risk.

Poor Quality Attribute Definitions ⇒ Higher Project Risk

Interoperability – A greater need for interoperability between your program and other programs and products introduces challenges for both the PMO and the supplier. The supplier must identify and manage all of the interprogram interoperability needs. The supplier must implement the interoperability requirements. Both of these factors introduce risk into the program. Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the best technical solution, providing some mitigation of this risk.

Greater Interoperability Needs ⇒ Higher Project Risk

Precedence – Developing an unprecedented system (a system unlike any that have been built before) is riskier than a precedented system (a system similar to previously built ones). Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the best technical solution, providing some mitigation of this risk.

Lack of System Precedence ⇒ Higher Project Risk

Quality Attribute Constraints – Rigid and extensive constraints on QAs such as reliability, supportability, security and so on, introduce challenge and risk into a program. Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the best technical solution, providing some mitigation of this risk.

Many Quality Attribute Constraints ⇒ Higher Project Risk

Technology Readiness – Reliance on immature technology may create program risk. Employing the broadest competition among the available suppliers elicits more ideas and provides some mitigation of this risk. Risk identification and mitigation may be stated source selection criteria used to help find the lowest risk approach.

Lack of Technology Readiness ⇒ Higher Project Risk

Legacy Considerations – Some systems must interoperate with legacy systems and some must be constructed with legacy components embedded within them. Yet others must provide operational performance identical to a legacy system. When compared with a “clean-sheet” design effort, all of these legacy considerations introduce technical challenges and risk into a program. Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the best technical solution to help provide some mitigation of this risk.

More Legacy Considerations ⇒ Higher Project Risk

COTS / GOTS / Reuse – Some systems are designed using COTS or GOTS software components. Some are designed to incorporate components that have been previously designed for other systems. While reusing these components can have significant benefit for the project, it also introduces risks, in that the system must accommodate the performance and the limitations of these components. Employing the broadest competition among the available suppliers elicits more ideas and helps the acquirer choose the most capable supplier and the best technical solution to provide some mitigation of this risk.

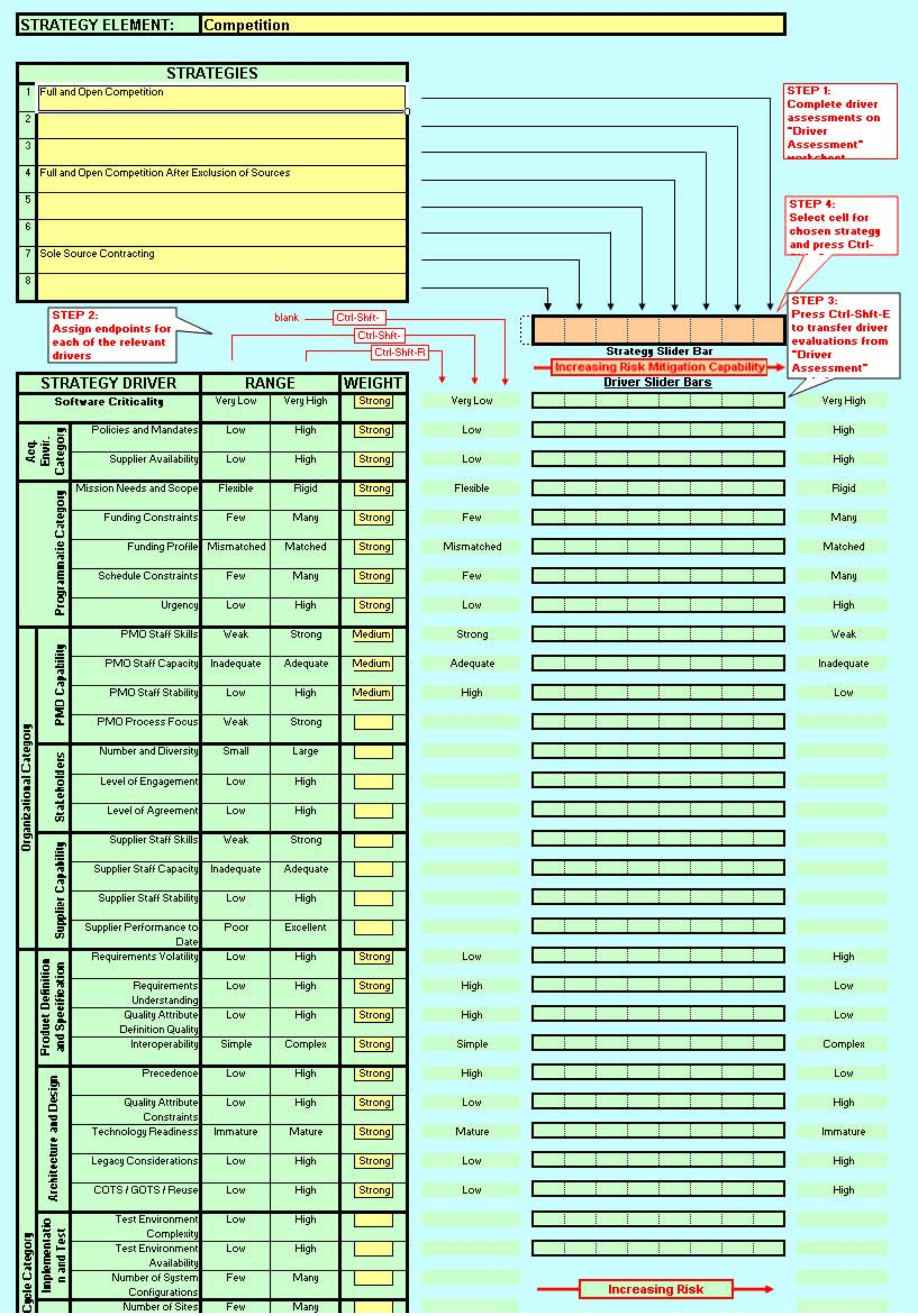
More COTS / GOTS / Reuse ⇒ Higher Project Risk

We now have an indication of the relationship between the strategy element and risk, and the relationships between the strategy drivers and risk, summarized in Table 16.

Table 16: Driver-to-Strategy Element Mapping: Competition

	Low Risk	**High Risk**
Strategy Element Risk Mitigation Capabilities		
Business Considerations: Competition	Sole Source	Full and Open Competition
Strategy Driver Risk Generation		
Software Criticality	Very Low	Very High
Policies and Mandates	Low	High
Supplier Availability	High	Low
Mission Needs and Scope	Flexible	Rigid
Funding Constraints	Few	Many
Funding Profile	Matched	Mismatched
Schedule Constraints	Few	Many
Urgency	Low	High
PMO Staff Skills	Strong	Weak
PMO Staff Capacity	Adequate	Inadequate
PMO Staff Stability	High	Low
Requirements Volatility	Low	High
Requirements Understanding	High	Low
Quality Attribute Definition Quality	High	Low
Interoperability	Simple	Complex
Precedence	High	Low
Quality Attribute Constraints	Low	High
Technology Readiness	Mature	Immature
Legacy Considerations	Low	High
COTS / GOTS / Reuse	Low	High

We can express the content of Table 16 graphically using slider bars, as shown in Figure 19. This information enables us to determine the assignment of endpoints to the driver strategy bars, with the “Low Risk” endpoint assigned to the left end of the slider bar and the “High Risk” endpoint assigned to the right end of the slider bar. Figure 19 also captures the strength of the linkage between the driver and the strategy element; the linkages of Table 14 (i.e., Strong, Medium, Weak, or None) are shown in the column labeled “WEIGHT.”



STRATEGY ELEMENT:	Competition

	STRATEGIES
1	Full and Open Competition
2	
3	
4	Full and Open Competition After Exclusion of Sources
5	
6	
7	Sole Source Contracting
8	

STEP 1:
Complete driver assessments on "Driver Assessment" worksheet

STEP 4:
Select cell for chosen strategy and press Ctrl-

STEP 3:
Press Ctrl-Shft-E to transfer driver evaluations from "Driver Assessment"

STEP 2:
Assign endpoints for each of the relevant drivers

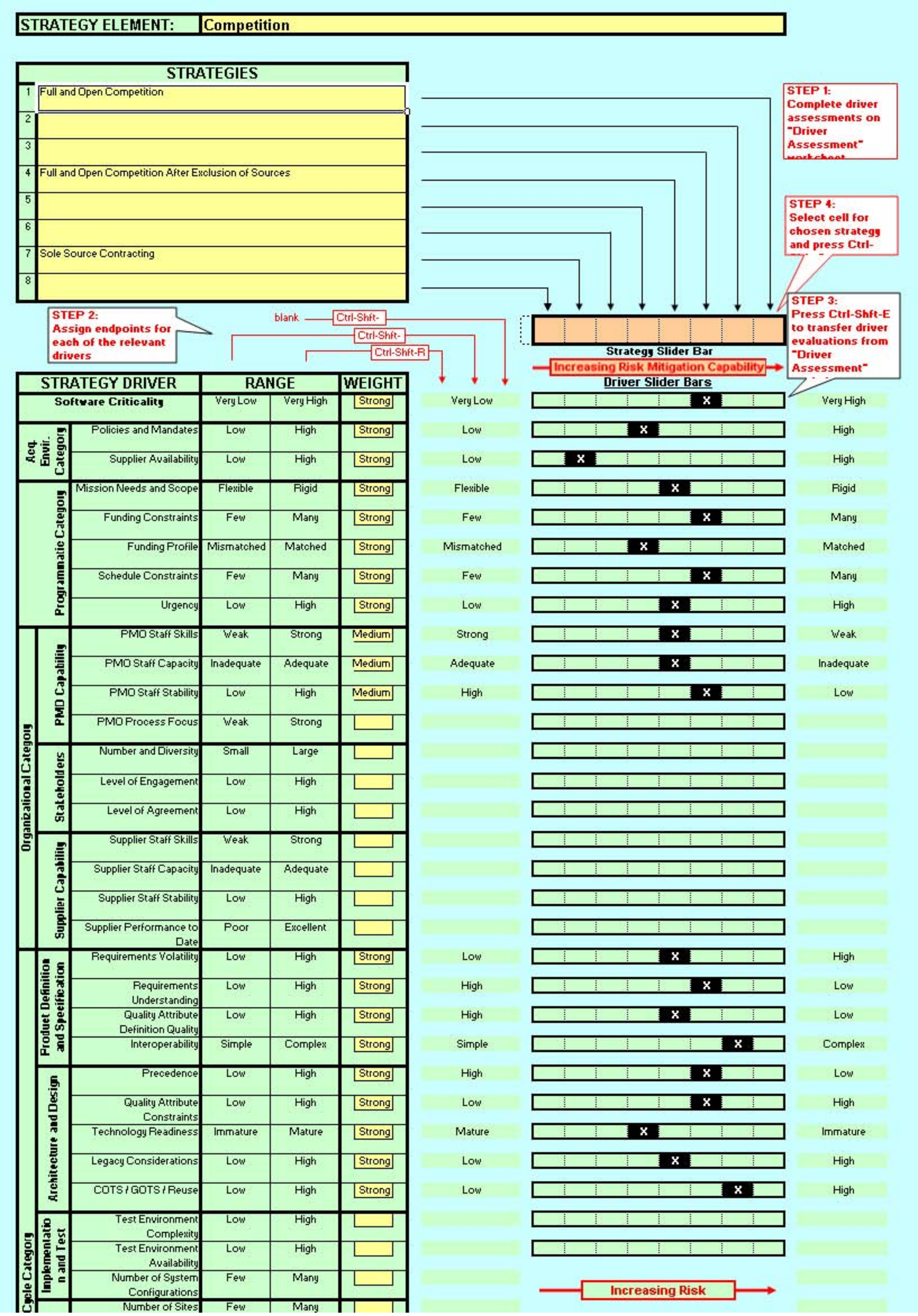
		STRATEGY DRIVER	RANGE		WEIGHT		
		Software Criticality	Very Low	Very High	Strong	Very Low	Very High
Acq. Envir. Category		Policies and Mandates	Low	High	Strong	Low	High
		Supplier Availability	Low	High	Strong	Low	High
Programmatic Category		Mission Needs and Scope	Flexible	Rigid	Strong	Flexible	Rigid
		Funding Constraints	Few	Many	Strong	Few	Many
		Funding Profile	Mismatched	Matched	Strong	Mismatched	Matched
		Schedule Constraints	Few	Many	Strong	Few	Many
		Urgency	Low	High	Strong	Low	High
Organizational Category	PMO Capability	PMO Staff Skills	Weak	Strong	Medium	Strong	Weak
		PMO Staff Capacity	Inadequate	Adequate	Medium	Adequate	Inadequate
		PMO Staff Stability	Low	High	Medium	High	Low
		PMO Process Focus	Weak	Strong			
	Stakeholders	Number and Diversity	Small	Large			
		Level of Engagement	Low	High			
		Level of Agreement	Low	High			
	Supplier Capability	Supplier Staff Skills	Weak	Strong			
		Supplier Staff Capacity	Inadequate	Adequate			
		Supplier Staff Stability	Low	High			
		Supplier Performance to Date	Poor	Excellent			
Cycle Category	Product Definition and Specification	Requirements Volatility	Low	High	Strong	Low	High
		Requirements Understanding	Low	High	Strong	High	Low
		Quality Attribute Definition Quality	Low	High	Strong	High	Low
		Interoperability	Simple	Complex	Strong	Simple	Complex
	Architecture and Design	Precedence	Low	High	Strong	High	Low
		Quality Attribute Constraints	Low	High	Strong	Low	High
		Technology Readiness	Immature	Mature	Strong	Mature	Immature
		Legacy Considerations	Low	High	Strong	Low	High
		COTS / GOTS / Reuse	Low	High	Strong	Low	High
	Implementation and Test	Test Environment Complexity	Low	High			
		Test Environment Availability	Low	High			
		Number of System Configurations	Few	Many			
		Number of Sites	Few	Many			

Figure 19: Slider Bar Set: Competition

Note that these relationships are subjective. For example, consider the “Quality Attribute Constraints” driver. We asserted that strong constraints on reliability, supportability, and so on, introduce challenge and complexity into the program, thereby creating a risk of cost overrun, schedule delays, and more. Thus, high “Quality Attribute Constraints” were equated with high risk. An alternative and equally valid perspective could be that strong constraints on reliability, supportability, and so on increase the likelihood of satisfying the mission needs and producing a fully capable product. In this case, high “Quality Attribute Constraints” would be equated with low risk.

5.4.3.7 Step 7 – Map Driver Evaluations to the Strategy Element

Now that the endpoints have been mapped for the selected strategy element, we can transfer these evaluations to the driver slider bars for the strategy element, as shown in Figure 20.



STRATEGY ELEMENT: Competition

STRATEGIES	
1	Full and Open Competition
2	
3	
4	Full and Open Competition After Exclusion of Sources
5	
6	
7	Sole Source Contracting
8	

STEP 1: Complete driver assessments on "Driver Assessment" worksheet

STEP 4: Select cell for chosen strategy and press Ctrl-

STEP 3: Press Ctrl-Shft-E to transfer driver evaluations from "Driver Assessment"

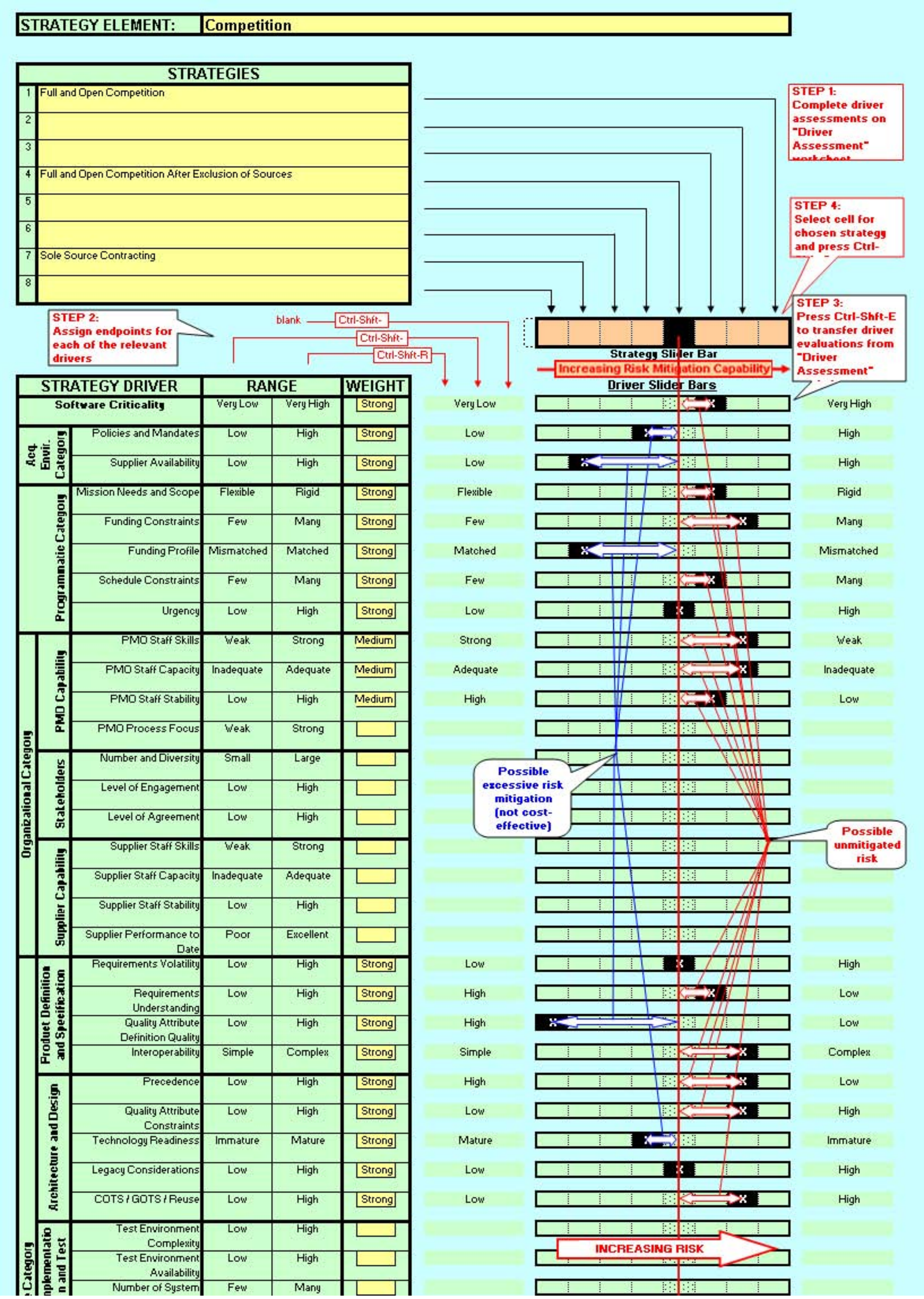
STEP 2: Assign endpoints for each of the relevant drivers

STRATEGY DRIVER			RANGE		WEIGHT		Driver Slider Bars								
Software Criticality			Very Low	Very High	Strong	Very Low						X			Very High
Acq. Envir. Category		Policies and Mandates	Low	High	Strong	Low				X					High
		Supplier Availability	Low	High	Strong	Low		X							High
Programmatic Category		Mission Needs and Scope	Flexible	Rigid	Strong	Flexible					X				Rigid
		Funding Constraints	Few	Many	Strong	Few						X			Many
		Funding Profile	Mismatched	Matched	Strong	Mismatched				X					Matched
		Schedule Constraints	Few	Many	Strong	Few						X			Many
		Urgency	Low	High	Strong	Low					X				High
Organizational Category	PMO Capability	PMO Staff Skills	Weak	Strong	Medium	Strong					X				Weak
		PMO Staff Capacity	Inadequate	Adequate	Medium	Adequate					X				Inadequate
		PMO Staff Stability	Low	High	Medium	High						X			Low
		PMO Process Focus	Weak	Strong											
	Stakeholders	Number and Diversity	Small	Large											
		Level of Engagement	Low	High											
		Level of Agreement	Low	High											
	Supplier Capability	Supplier Staff Skills	Weak	Strong											
		Supplier Staff Capacity	Inadequate	Adequate											
		Supplier Staff Stability	Low	High											
		Supplier Performance to Date	Poor	Excellent											
Cycle Category	Product Definition and Specification	Requirements Volatility	Low	High	Strong	Low					X				High
		Requirements Understanding	Low	High	Strong	High						X			Low
		Quality Attribute Definition Quality	Low	High	Strong	High					X				Low
		Interoperability	Simple	Complex	Strong	Simple							X		Complex
	Architecture and Design	Precedence	Low	High	Strong	High						X			Low
		Quality Attribute Constraints	Low	High	Strong	Low						X			High
		Technology Readiness	Immature	Mature	Strong	Mature				X					Immature
		Legacy Considerations	Low	High	Strong	Low					X				High
		COTS / GOTS / Reuse	Low	High	Strong	Low							X		High
	Implementatio n and Test	Test Environment Complexity	Low	High											
		Test Environment Availability	Low	High											
		Number of System Configurations	Few	Many											
		Number of Sites	Few	Many											

Figure 20: Driver Evaluations Mapped to Strategy Element

5.4.3.8 Step 8 – Choose Strategy

As shown in Figure 20, most of the strategy drivers are marked between the middle and the right (high risk) side of the slider bars. This indicates that the perceived amount of risk from these drivers is moderate to high. Our goal is to choose a strategy that mitigates the risks as much as possible. As an example, we will choose the middle strategy, Full and Open Competition After Exclusion of Sources. As shown in Figure 21, this strategy provides some risk mitigation for many, but not all of the drivers. Drivers with evaluations to the right of the strategy represent risks that the strategy element is not mitigating to its fullest extent. These risks need to be addressed through other risk management activities. On the other hand, many of the driver evaluations are marked to the left of the chosen strategy. We could interpret this to mean that the risks from these drivers are “over-mitigated.” While this may be beneficial from a risk management perspective, it may not be cost-effective; you may be choosing a more expensive strategy than is warranted for the level of risk in your program.



STRATEGY ELEMENT:	Competition

	STRATEGIES
1	Full and Open Competition
2	
3	
4	Full and Open Competition After Exclusion of Sources
5	
6	
7	Sole Source Contracting
8	

STEP 1:
Complete driver assessments on "Driver Assessment" worksheet

STEP 4:
Select cell for chosen strategy and press Ctrl-

STEP 3:
Press Ctrl-Shft-E to transfer driver evaluations from "Driver Assessment"

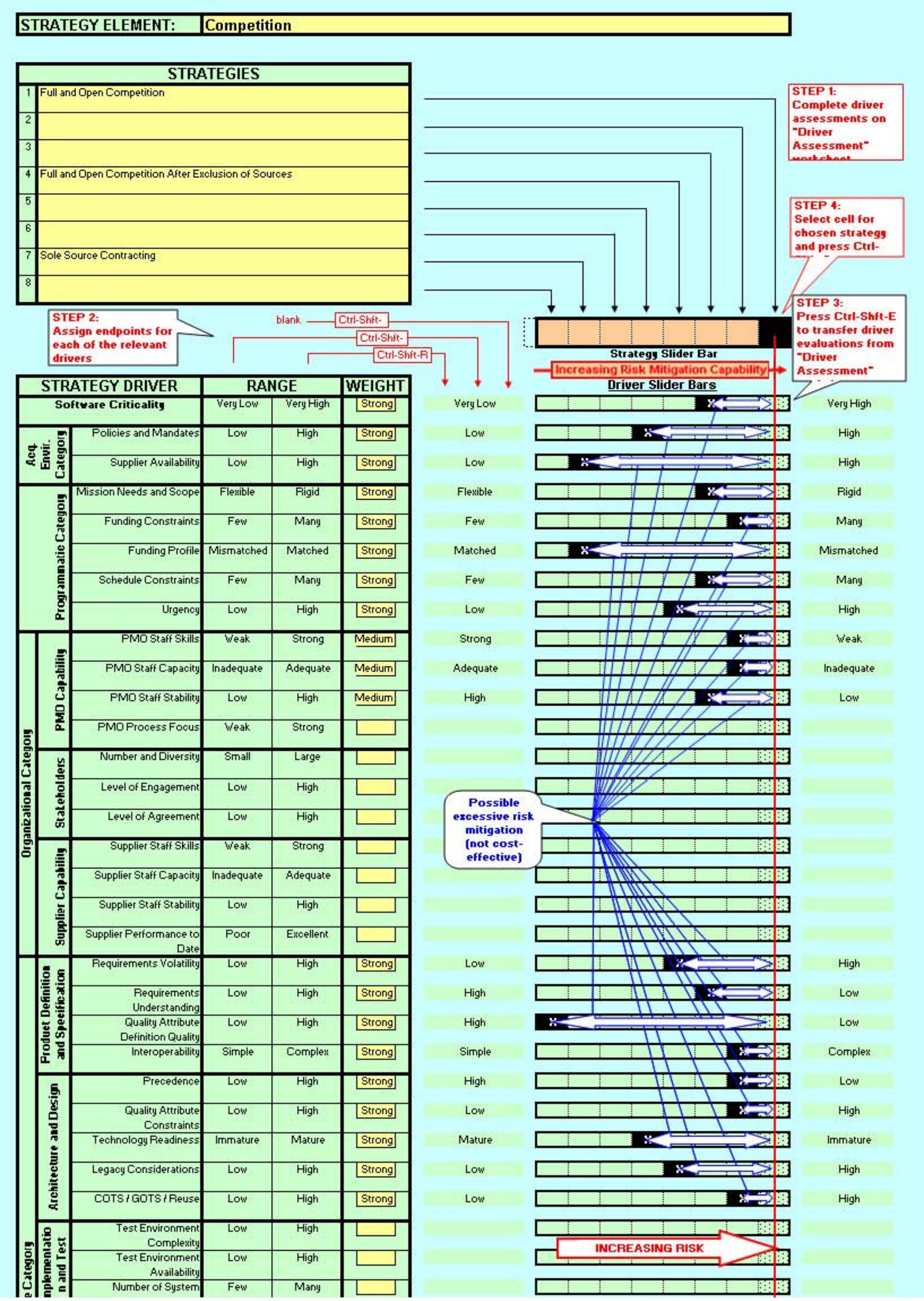
STEP 2:
Assign endpoints for each of the relevant drivers

blank
Ctrl-Shft-
Ctrl-Shft-
Ctrl-Shft-R
Strategy Slider Bar
Increasing Risk Mitigation Capability
Driver Slider Bars
Possible excessive risk mitigation (not cost-effective)
Possible unmitigated risk
INCREASING RISK

		STRATEGY DRIVER	RANGE		WEIGHT		
		Software Criticality	Very Low	Very High	Strong	Very Low	Very High
Acq. Envir. Category		Policies and Mandates	Low	High	Strong	Low	High
		Supplier Availability	Low	High	Strong	Low	High
Programmatic Category		Mission Needs and Scope	Flexible	Rigid	Strong	Flexible	Rigid
		Funding Constraints	Few	Many	Strong	Few	Many
		Funding Profile	Mismatched	Matched	Strong	Matched	Mismatched
		Schedule Constraints	Few	Many	Strong	Few	Many
		Urgency	Low	High	Strong	Low	High
Organizational Category	PMO Capability	PMO Staff Skills	Weak	Strong	Medium	Strong	Weak
		PMO Staff Capacity	Inadequate	Adequate	Medium	Adequate	Inadequate
		PMO Staff Stability	Low	High	Medium	High	Low
		PMO Process Focus	Weak	Strong			
	Stakeholders	Number and Diversity	Small	Large			
		Level of Engagement	Low	High			
		Level of Agreement	Low	High			
	Supplier Capability	Supplier Staff Skills	Weak	Strong			
		Supplier Staff Capacity	Inadequate	Adequate			
		Supplier Staff Stability	Low	High			
		Supplier Performance to Date	Poor	Excellent			
	Product Definition and Specification	Requirements Volatility	Low	High	Strong	Low	High
		Requirements Understanding	Low	High	Strong	High	Low
		Quality Attribute Definition Quality	Low	High	Strong	High	Low
		Interoperability	Simple	Complex	Strong	Simple	Complex
	Architecture and Design	Precedence	Low	High	Strong	High	Low
		Quality Attribute Constraints	Low	High	Strong	Low	High
		Technology Readiness	Immature	Mature	Strong	Mature	Immature
		Legacy Considerations	Low	High	Strong	Low	High
		COTS / GOTS / Reuse	Low	High	Strong	Low	High
e Category	nplementatio n and Test	Test Environment Complexity	Low	High			
		Test Environment Availability	Low	High			
		Number of System	Few	Many			

Figure 21: Competition Strategy #1

Alternatively, if we choose a Full and Open Competition strategy, the slider bar set appears as shown in Figure 22. The strategy selection means that the maximum amount of risk mitigation available from the strategy element has been applied to the risk arising from all of the drivers. It does not mean that the risk is fully mitigated. Now, all of the drivers show excessive mitigation, a situation that may not be cost effective.



STRATEGY ELEMENT:	Competition

STRATEGIES	
1	Full and Open Competition
2	
3	
4	Full and Open Competition After Exclusion of Sources
5	
6	
7	Sole Source Contracting
8	

STEP 1:
Complete driver assessments on "Driver Assessment" worksheet

STEP 4:
Select cell for chosen strategy and press Ctrl-

STEP 3:
Press Ctrl-Shft-E to transfer driver evaluations from "Driver Assessment"

STEP 2:
Assign endpoints for each of the relevant drivers

Possible excessive risk mitigation (not cost-effective)

		STRATEGY DRIVER	RANGE		WEIGHT		
		Software Criticality	Very Low	Very High	Strong	Very Low	Very High
Acq. Envir. Category		Policies and Mandates	Low	High	Strong	Low	High
		Supplier Availability	Low	High	Strong	Low	High
Programmatic Category		Mission Needs and Scope	Flexible	Rigid	Strong	Flexible	Rigid
		Funding Constraints	Few	Many	Strong	Few	Many
		Funding Profile	Mismatched	Matched	Strong	Matched	Mismatched
		Schedule Constraints	Few	Many	Strong	Few	Many
		Urgency	Low	High	Strong	Low	High
Organizational Category	PMO Capability	PMO Staff Skills	Weak	Strong	Medium	Strong	Weak
		PMO Staff Capacity	Inadequate	Adequate	Medium	Adequate	Inadequate
		PMO Staff Stability	Low	High	Medium	High	Low
		PMO Process Focus	Weak	Strong			
	Stakeholders	Number and Diversity	Small	Large			
		Level of Engagement	Low	High			
		Level of Agreement	Low	High			
	Supplier Capability	Supplier Staff Skills	Weak	Strong			
		Supplier Staff Capacity	Inadequate	Adequate			
		Supplier Staff Stability	Low	High			
		Supplier Performance to Date	Poor	Excellent			
e Category	Product Definition and Specification	Requirements Volatility	Low	High	Strong	Low	High
		Requirements Understanding	Low	High	Strong	High	Low
		Quality Attribute Definition Quality	Low	High	Strong	High	Low
		Interoperability	Simple	Complex	Strong	Simple	Complex
	Architecture and Design	Precedence	Low	High	Strong	High	Low
		Quality Attribute Constraints	Low	High	Strong	Low	High
		Technology Readiness	Immature	Mature	Strong	Mature	Immature
		Legacy Considerations	Low	High	Strong	Low	High
		COTS / GOTS / Reuse	Low	High	Strong	Low	High
	nplementatio n and Test	Test Environment Complexity	Low	High			
		Test Environment Availability	Low	High			
		Number of System	Few	Many			

Figure 22: Competition Strategy #2

5.4.3.9 Step 9 – Identify Residual Risks

The strategy selection performed in Step 5 does not necessarily provide complete mitigation of the risks arising from the drivers. Choosing a strategy “to the right” of the strategy driver evaluation does not mean that the risk is fully mitigated. It merely means that the maximum amount of risk mitigation available from the strategy element has been applied to that risk. Additional risk mitigation actions may be required through the program’s risk mitigation process.

Thus, after the strategy is chosen, the PMO must examine the program to identify and manage the residual risks.

5.4.4 Evaluating an Existing Strategy

For a program with an acquisition strategy already in execution, slider bars can be used to perform a strategy analysis. Other than Step 8, the steps are similar to those outlined in Section 5.4.3.

1. Define the acquisition objectives.
2. Identify and evaluate drivers the factors that drive the program.
3. Decompose the strategy into individual strategy elements.
4. For the selected strategy element, identify the potential strategic choices, and rank them in order of their risk mitigation capabilities for your particular program.
5. Evaluate the strategy drivers for the program to identify those that influence the strategy element through the introduction of risk that may be mitigated by strategy element.
6. Define the relationship between the risk generated by the strategy driver and the risk mitigation capabilities of the strategy element.
7. Map the driver evaluations from Step 1 to the strategy element using the slider bars.
8. Indicate current strategy.
9. Identify residual risks.

Perform Steps 1–7 exactly as described in Section 5.4.3. For Step 8, indicate the previously defined program strategy that is currently in execution on the strategy element slider bar. For Step 9, evaluate the residual risks resulting from the strategy drivers “to the right” of the chosen strategy, as in Section 5.4.3. These risks must then be addressed in the program’s risk management activities.

Conclusion

Developing an acquisition strategy is a process that is not well understood. In most cases, it is performed in an ad hoc manner, without due consideration of the internal and external factors driving the project. In crafting an acquisition strategy, the program manager must balance many conflicting objectives, choosing a strategy that addresses some and ignores others.

The research results presented in this report support a more systematic approach to reasoning about software risk on a program. The methods and techniques presented contribute to the work focused on developing an acquisition strategy from a sound, systems engineering approach.

The report is designed to help acquisition planners more systematically

- profile conditions that impact program risk (drivers)
- identify sources of potential software risk early in the program and throughout the program life cycle
- develop an acquisition strategy by decomposing it into the strategy elements that compose it and addressing the elements individually and then collectively
- reason about their acquisition strategy’s ability to mitigate software risk in an ongoing manner

One of the key methods proposed can be used to develop an acquisition strategy by evaluating a program’s drivers, or it can be used to evaluate an existing acquisition strategy given the program’s drivers.

To develop a strategy acquisition planners would

1. Define the objectives of the acquisition.
2. Identify and evaluate the factors that drive the program.
3. Decompose the strategy into individual strategy elements.
4. For the selected strategy elements, identify the potential strategic choices and rank them in order of their risk mitigation capabilities for you program.
5. Evaluate the strategy drivers for the program to identify those that influence the strategy element through the introduction of risk that may be mitigated by the strategy element.
6. Define the relationship between the risk generated by the strategy drive and the risk mitigation capabilities of the strategy element.
7. Map the driver evaluations from Step 2 to the strategy element using the slider bars.

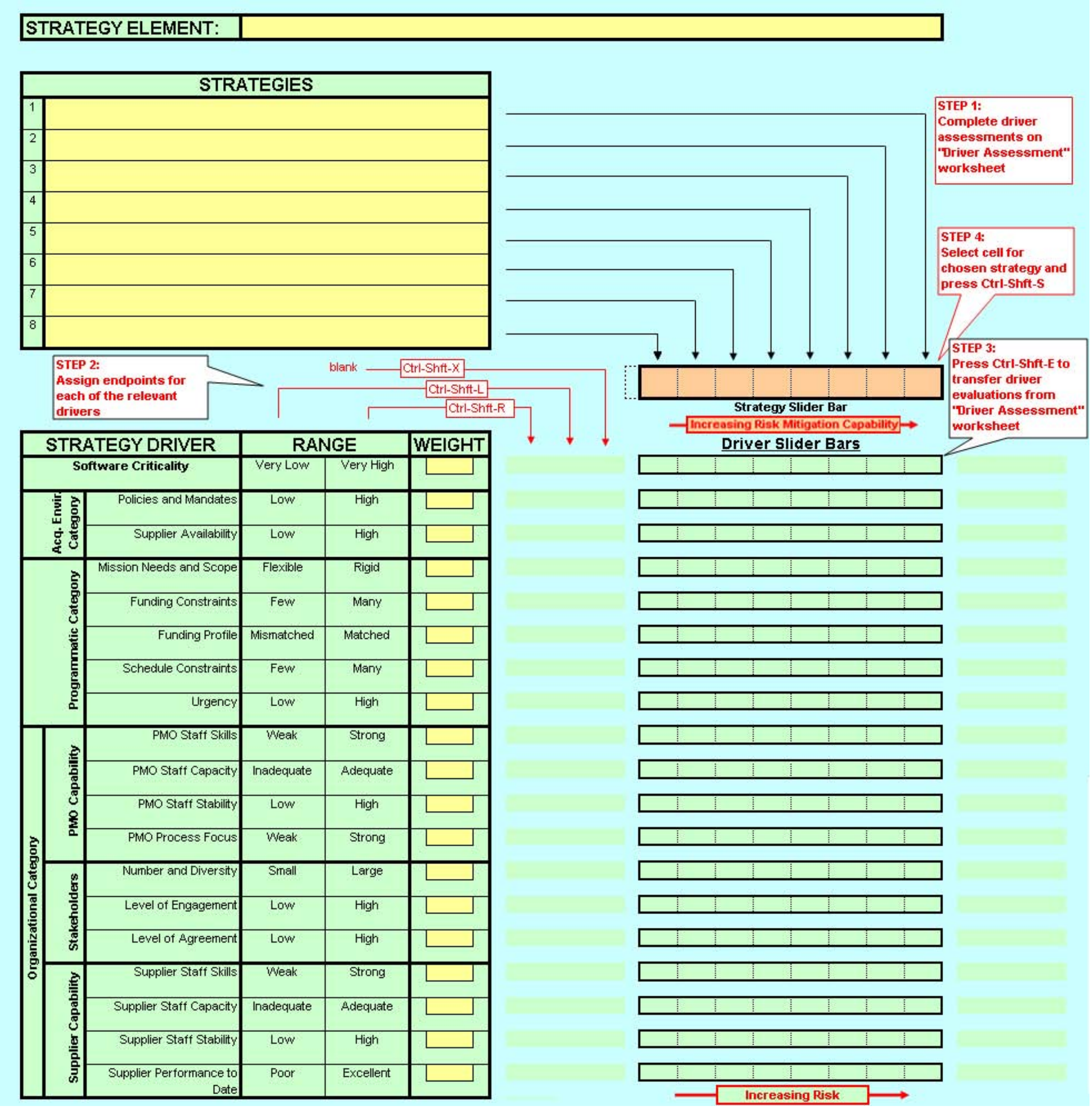
8. Choose the strategy that best mitigates the risk elements.
9. Identify residual risks.

For a program with an acquisition strategy already in execution, slider bars can be used to perform a strategy analysis. Steps 1–7 are performed exactly as described in Section 5.4.3. For Step 8, indicate the previously defined program strategy that is currently in execution on the strategy element slider bar. For Step 9, evaluate the residual risks resulting from the strategy drivers "to the right" of the chosen strategy, as described in Section 5.4.3. These risks must then be addressed in the program's risk management activities.

The ASDT is provided to help acquisition planners work through the method and techniques. Acquisition planners can apply the method and techniques discussed in this report without using the ASDT. The ASDT is provided so that acquisition organizations do not have to develop templates from scratch. The ASDT is available for download on the SEI Publications Web site at http://www.sei.cmu.edu/publications/publications.html.

The research presented in this report focused on identifying and mitigating software risk in a program during the acquisition planning process. However, the methods and techniques we present can be applied to acquisition planning areas other than software risk.

Appendix A Templates for Software Driver Slider Bars



STRATEGY ELEMENT:

STRATEGIES

1
2
3
4
5
6
7
8

STEP 1:
Complete driver assessments on "Driver Assessment" worksheet

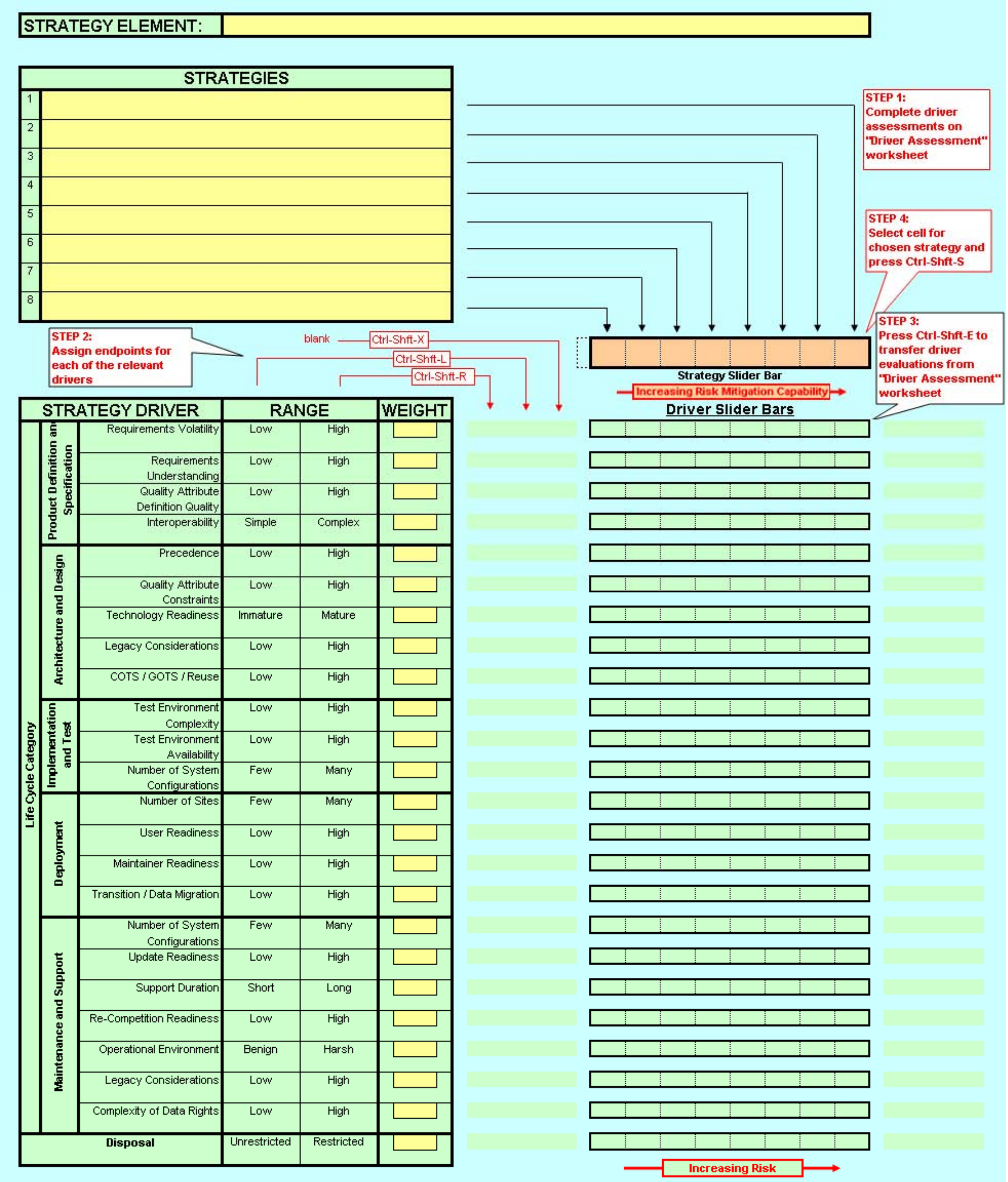
STEP 4:
Select cell for chosen strategy and press Ctrl-Shft-S

STEP 2:
Assign endpoints for each of the relevant drivers

STEP 3:
Press Ctrl-Shft-E to transfer driver evaluations from "Driver Assessment" worksheet

STRATEGY DRIVER			RANGE		WEIGHT
Software Criticality			Very Low	Very High	
Acq. Envir. Category		Policies and Mandates	Low	High	
		Supplier Availability	Low	High	
Programmatic Category		Mission Needs and Scope	Flexible	Rigid	
		Funding Constraints	Few	Many	
		Funding Profile	Mismatched	Matched	
		Schedule Constraints	Few	Many	
		Urgency	Low	High	
Organizational Category	PMO Capability	PMO Staff Skills	Weak	Strong	
		PMO Staff Capacity	Inadequate	Adequate	
		PMO Staff Stability	Low	High	
		PMO Process Focus	Weak	Strong	
	Stakeholders	Number and Diversity	Small	Large	
		Level of Engagement	Low	High	
		Level of Agreement	Low	High	
	Supplier Capability	Supplier Staff Skills	Weak	Strong	
		Supplier Staff Capacity	Inadequate	Adequate	
		Supplier Staff Stability	Low	High	
		Supplier Performance to Date	Poor	Excellent	

Figure 23: Strategy Slider Bar Template



STRATEGY ELEMENT:

STRATEGIES	
1	
2	
3	
4	
5	
6	
7	
8	

STEP 1: Complete driver assessments on "Driver Assessment" worksheet

STEP 2: Assign endpoints for each of the relevant drivers

STEP 3: Press Ctrl-Shft-E to transfer driver evaluations from "Driver Assessment" worksheet

STEP 4: Select cell for chosen strategy and press Ctrl-Shft-S

STRATEGY DRIVER			RANGE		WEIGHT
Life Cycle Category	Product Definition an Specification	Requirements Volatility	Low	High	
		Requirements Understanding	Low	High	
		Quality Attribute Definition Quality	Low	High	
		Interoperability	Simple	Complex	
	Architecture and Design	Precedence	Low	High	
		Quality Attribute Constraints	Low	High	
		Technology Readiness	Immature	Mature	
		Legacy Considerations	Low	High	
		COTS / GOTS / Reuse	Low	High	
	Implementation and Test	Test Environment Complexity	Low	High	
		Test Environment Availability	Low	High	
		Number of System Configurations	Few	Many	
	Deployment	Number of Sites	Few	Many	
		User Readiness	Low	High	
		Maintainer Readiness	Low	High	
		Transition / Data Migration	Low	High	
	Maintenance and Support	Number of System Configurations	Few	Many	
		Update Readiness	Low	High	
		Support Duration	Short	Long	
		Re-Competition Readiness	Low	High	
		Operational Environment	Benign	Harsh	
		Legacy Considerations	Low	High	
		Complexity of Data Rights	Low	High	
Disposal			Unrestricted	Restricted	

Figure 24: Strategy Slider Bar Template (continued)

Appendix B Acronyms

IA	Information Assurance
AP	Acquisition Plan
ASDT	Acquisition Strategy Development Tool
BEA	Business Enterprise Architecture
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
CBT	Computer-Based Training
CDD	Capabilities Development Document
CLS	Contractor Logistics Support
CMMI	Capability Maturity Model Integration
CMMI-AM	Capability Maturity Model Integration – Acquisition Module
COTS	Commercial Off-the-Shelf
CoN	Certificate of Networthiness
CtO	Certificate to Operate
DAU	Defense Acquisition University
DL	Distance Learning
DM	Data Management
DoD	Department of Defense
ESOH	Environment, Safety, and Occupational Health
FAR	Federal Acquisition Regulation
FFP	Firm-Fixed-Price
F/OS	Free/Open Source

GFE	Government Furnished Equipment
GFI	Government Furnished Information
GOTS	Government Off-the-Shelf
HSI	Human Systems Integration
IA	Information Assurance
IFB	Invitation for Bid
JCIDS	Joint Capabilities Integration and Development System
JROC	Joint Requirements Oversight Council
JTA	Joint Technical Architecture
MBT	Main Battle Tank
MOSA	Modular Open Systems Approach
ORD	Operational Requirements Document
PBL	Performance Based Logistics
PCA	Permanent Change of Assignment
PCS	Permanent Change of Station
PM	Program Manager
PMO	Program Management Office
QA	Quality Attribute
RFI	Request for Information
RFP	Request for Proposal
RFQ	Request for Quotation
SETA	Systems Engineering and Technical Assistance
SLOC	Source Lines of Code
SOO	Statement of Objectives
SOW	Statement of Work

SIPRNet	Secret Internet Protocol Router Network
TEMP	Test and Evaluation Master Plan
UAGV	Unmanned Autonomous Ground Vehicle
XML	Extensible Markup Language

Bibliography

URLs are valid as of the publication date of this document.

[Bernard 05] Bernard, Tom; Gallagher, Brian; Bate, Roger; & Wilson, Hal. *CMMI Acquisition Module (CMMI-AM), Version 1.1* (CMU/SEI-2005-TR-011, ADA441245). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2005. http://www.sei.cmu.edu/publications/documents/05.reports/05tr011.html.

[Brownsword 04] Brownsword, Lisa L.; Carney, David J.; Fisher, David; Lewis, Grace; Meyers, Craig; Morris, Edwin J., Place, Patrick R. H.; Smith, James; & Wrage, Lutz. *Current Perspectives on Interoperability* (CMU/SEI-2004-TR-009). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04tr009.html.

[Carney 03] Carney, David J.; Morris, Edwin J.; & Place, Patrick R. H. *Identifying Commercial Off-the-Shelf (COTS) Product Risks: The COTS Usage Risk Evaluation* (CMU/SEI-2003-TR-023, ADA418382). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2003. http://www.sei.cmu.edu/publications/documents/03.reports/03tr023.html.

[DAU 03] Defense Acquisition University. *Defense Acquisition Definitions and Terms, 11th Edition*. Fort Belvoir, VA: Defense Acquisition University Press, September 2003.

[DAU 04] Defense Acquisition University (DAU). *Defense Acquisition Guidebook*. http://akss.dau.mil/dag/DoD5000.asp?view=document (2004).

[DoD 96] Department of Defense. *Mandatory Procedures for Major Defense Acquisition Programs (MDAPS) and Major Automated Information System (MAIS) Acquisition Programs*. 1996.

[DoD 03a] Department of Defense. *DoD Directive The Defense Acquisition System (DoD 5000.1)*. http://www.dtic.mil/whs/directives/corres/pdf2/d50001p.pdf (2003).

[DoD 03b] Department of Defense. *DoD Instruction Operation of the Defense Acquisition System (DoDI 5000.2)*. http://www.dtic.mil/whs/directives/corres/pdf2/i50002p.pdf (2003).

[Department of the Navy 01] Department of the Navy. *Acquisition Strategy Decision Guide*. http://acquisition.navy.mil/aosfiles/tools/asdg/ (2001).

[Dorofee 96] Dorofee, Audrey; Walker, Julie, Alberts, Christopher; Higuera, Ronald; Murphy, Richard; & Williams, Ray. *Continuous Risk Management Guidebook*. Pittsburgh, PA: Carnegie Mellon University, 1996.

[DSB 00] Defense Science Board. *Report of the Defense Science Board Task Force On Defense Software.* http://www.acq.osd.mil/dsb/reports/defensesoftware.pdf (2000).

[GAO 04a] United States Government Accountability Office. *Stronger Management Practices Are Needed to Improve DOD's Software-Intensive Weapon Acquisitions*. http://www.gao.gov/new.items/d04393.pdf (March 2004).

[GAO 04b] United States Government Accountability Office. *Knowledge of Software Suppliers Needed to Manage Risks*. http://www.gao.gov/new.items/d04678.pdf (May 2004).

[GAO 05] United States Government Accountability Office. *Future Combat Systems Challenges and Prospects for Success*. http://www.gao.gov/new.items/d05442t.pdf (March 2005).

[Leffingwell 03] Leffingwell, Dean & Widrig, Don. *Managing Software Requirements – Second Edition*. Boston, MA: Addison-Wesley, 2003.

[Marciniak 05] Marciniak, John J., ed. *Encyclopedia of Software Engineering.* http://www3.interscience.wiley.com/cgi-bin/mrwhome/104554786/HOME (2005).

[McGarry 03] McGarry, John. *Systemic Analysis of Software Intensive System Acquisition Issues*. http://www.psmsc.com/Downloads/Other/McGarrySTC2003SystemicPresentationVer2.pdf (2003).

[Merriam-Webster 05] Merriam-Webster. *Merriam-Webster Online Dictionary.* http://www.m-w.com (2005).

[Morris 04] Morris, Edwin; Levine, Linda; Meyers, Craig; Place, Patrick R.H.; & Plakosh, Daniel. *System of Systems Interoperability Final Report.* (CMU/SEI-2004-TR-004). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2004. http://www.sei.cmu.edu/publications/documents/04.reports/04tr004.html.

[Nelson 99] Nelson, Maj. Mike; Clark, James; & Spurlock, Martha Ann. “Curing the Software Requirements and Cost Estimating Blues.” *Program Manager* (November–December 1999): 54–60.

[OUSD AT&L 01] Office of the Under Secretary of Defense for Acquisition, Technology and Logistics. *Product Support: A Program Manager’s Guide to Buying Performance.* http://www.acq.osd.mil/log/logistics_materiel_readiness/organizations/lpp/assetts/product_support/new_prd_spt_gde/PSGuide-nov01.pdf (2001).

[Sherry 96] Sherry, Lorraine. “Issues in Distance Learning.” *International Journal of Educational Telecommunications, 1* (4), (1996): 337–365. http://carbon.cudenver.edu/~lsherry/pubs/issues.html.

[Standish 03] The Standish Group International, Inc. *Chaos Chronicles, Version 3.0*. Yarmouth, MA: The Standish Group, 2003.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE August 2006	3. REPORT TYPE AND DATES COVERED Final

4. TITLE AND SUBTITLE Techniques for Developing an Acquisition Strategy by Profiling Software Risks	5. FUNDING NUMBERS FA8721-05-C-0003

6. AUTHOR(S)

Mary Catherine Ward, Joseph P. Elm, Susan Kushner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	8. PERFORMING ORGANIZATION REPORT NUMBER CMU/SEI-2006-TR-002
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	10. SPONSORING/MONITORING AGENCY REPORT NUMBER ESC-TR-2006-002

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT Unclassified/Unlimited, DTIC, NTIS	12B DISTRIBUTION CODE

13. ABSTRACT (MAXIMUM 200 WORDS)

The goal of acquisition planning is to create a roadmap that a program can follow to maximize its chances of successfully fielding a system that meets users' needs within cost and on schedule. Developing an acquisition strategy is a key component of acquisition planning that provides a means of addressing risks through the program structure. Programs need structured ways to reason about software risks, formulate acquisition strategies to mitigate software risk, and evaluate their current acquisition strategy in an ongoing, systematic manner.

This report introduces a taxonomy of strategy drivers and strategy elements and provides a method for performing a comparative analysis of the strategy drivers and the resulting strategic choices for the elements. The primary audience for this technical report and the accompanying Excel-based tool is program managers of government acquisition programs. The main prerequisite for successfully using this information is working knowledge of government acquisition practices.

14. SUBJECT TERMS acquisition, assurance, COTS, information security, interoperability, security, sustainment, risk, requirements, taxonomy, testing, systems engineering, strategic planning	15. NUMBER OF PAGES 136

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL